# Using your existing Java knowledge to create cloud happy applications

Eero Arvonen
Architect

Red Hat

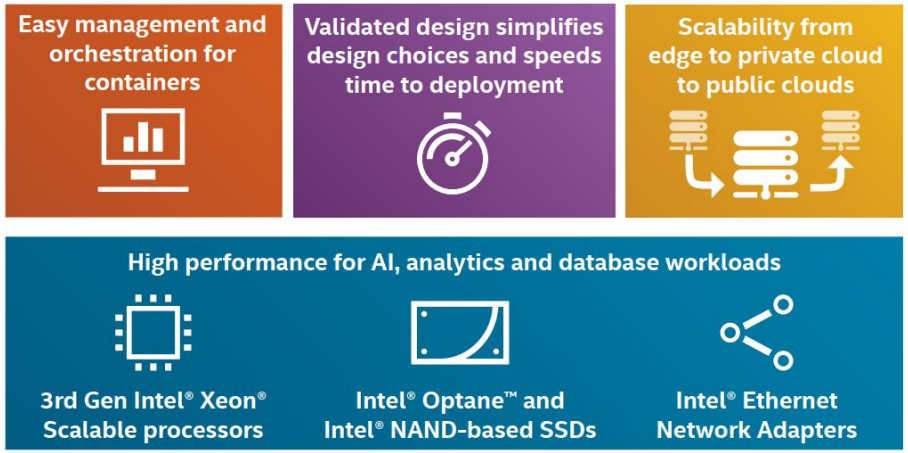
# Red Hat OpenShift Reference Architecture

Joint Red Hat and Intel OpenShift Reference Architecture

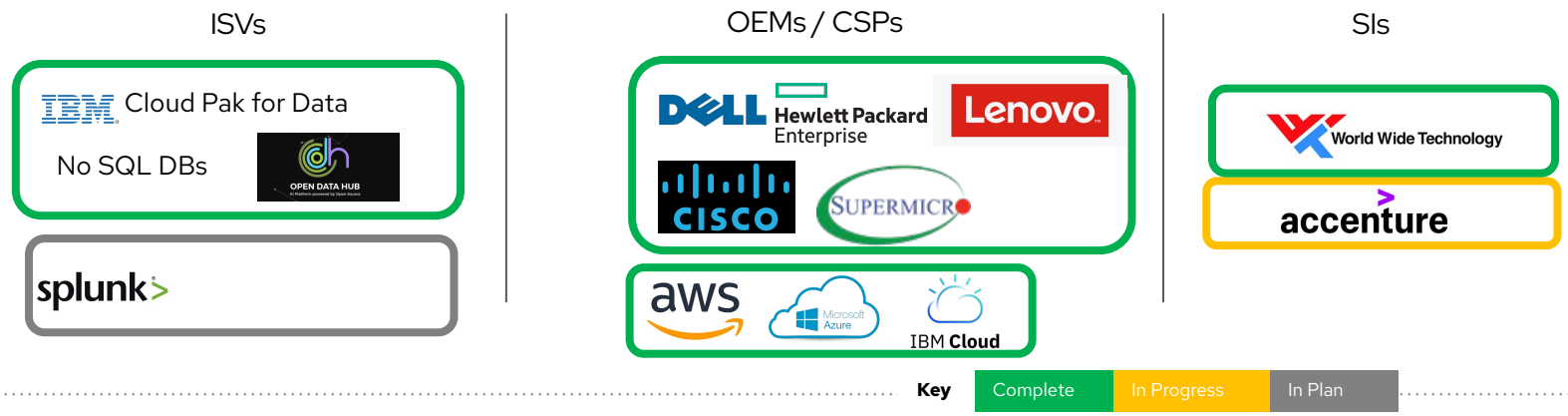
## Solution overview

**Summary:** The RA enables deployment of performant and low-latency container-based workloads onto different footprints, such as bare metal, virtual, private cloud, public cloud, or a combination of these, in either a centralized data center or at the edge

**Purpose:** A general purpose OpenShift reference architecture to showcase the best of Intel and Red Hat products with key workloads




## Solution ecosystem




## Intel enabling status

- Intel® Xeon (2nd Gen – Cascade Lake, 3rd Gen – Ice Lake)
- Intel Optane (PMEM, SSD); Columbiaville

## Collateral

- Intel OpenShift RA for 4.6
- Intel OpenShift Solution Brief for 4.6
- Red Hat: OpenShift Ref Arch – Multiple OEMs
- Dell: OpenShift Offering
- HPE: OpenShift Offering
- Cisco: OpenShift Offering
- Lenovo: OpenShift Offering
- Supermicro: OpenShift Offering
- Penguin Computing: OpenShift Offering



Red Hat | intel.

# Agenda

- Introduction
- New architectures drives new technology needs
- Approach to meet new needs
- Summary


Red Hat

# 9M+

Java developers worldwide

**#1 Availability of developers**

# 90%

of the Fortune 500 are using Java

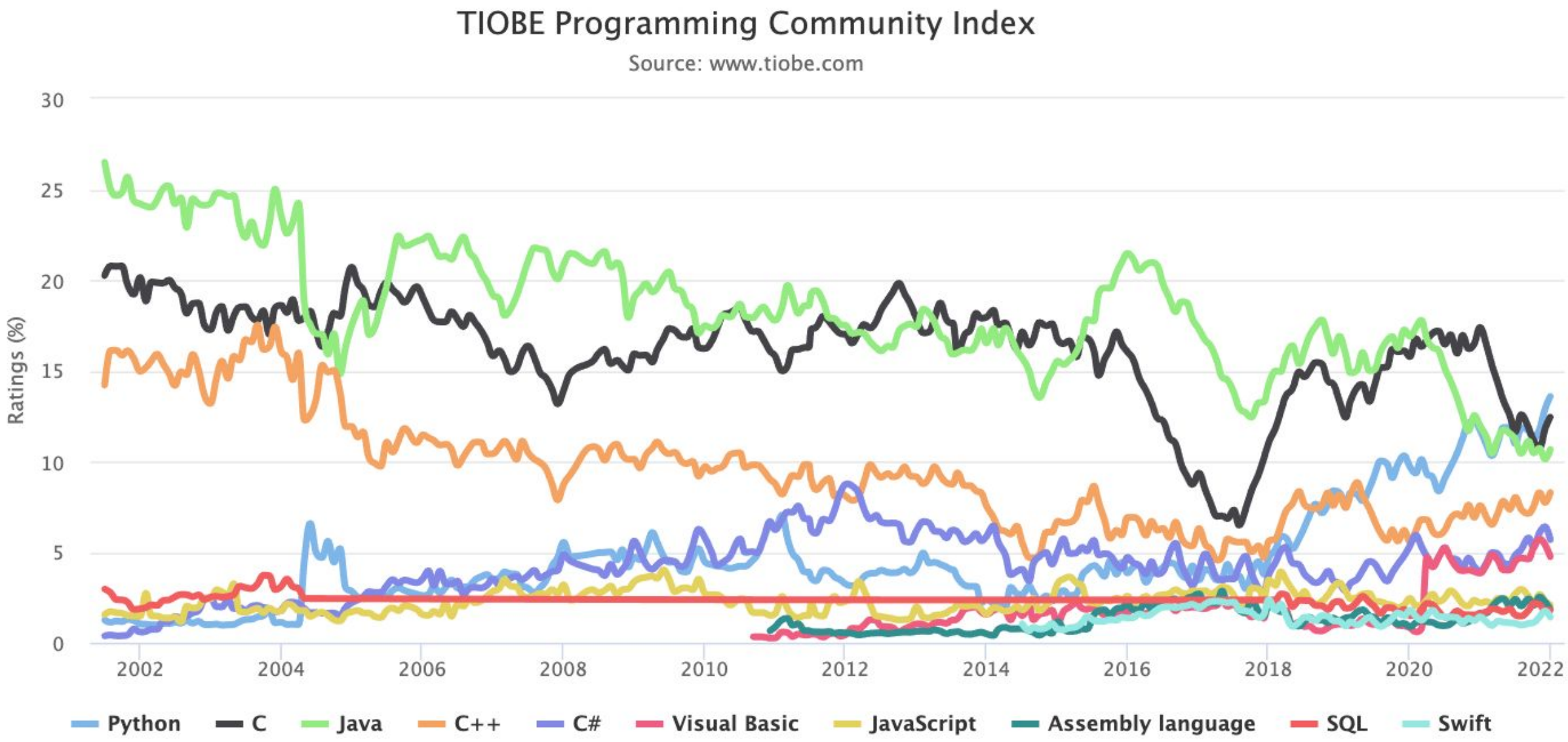
**#2 Specifications**

# 40%

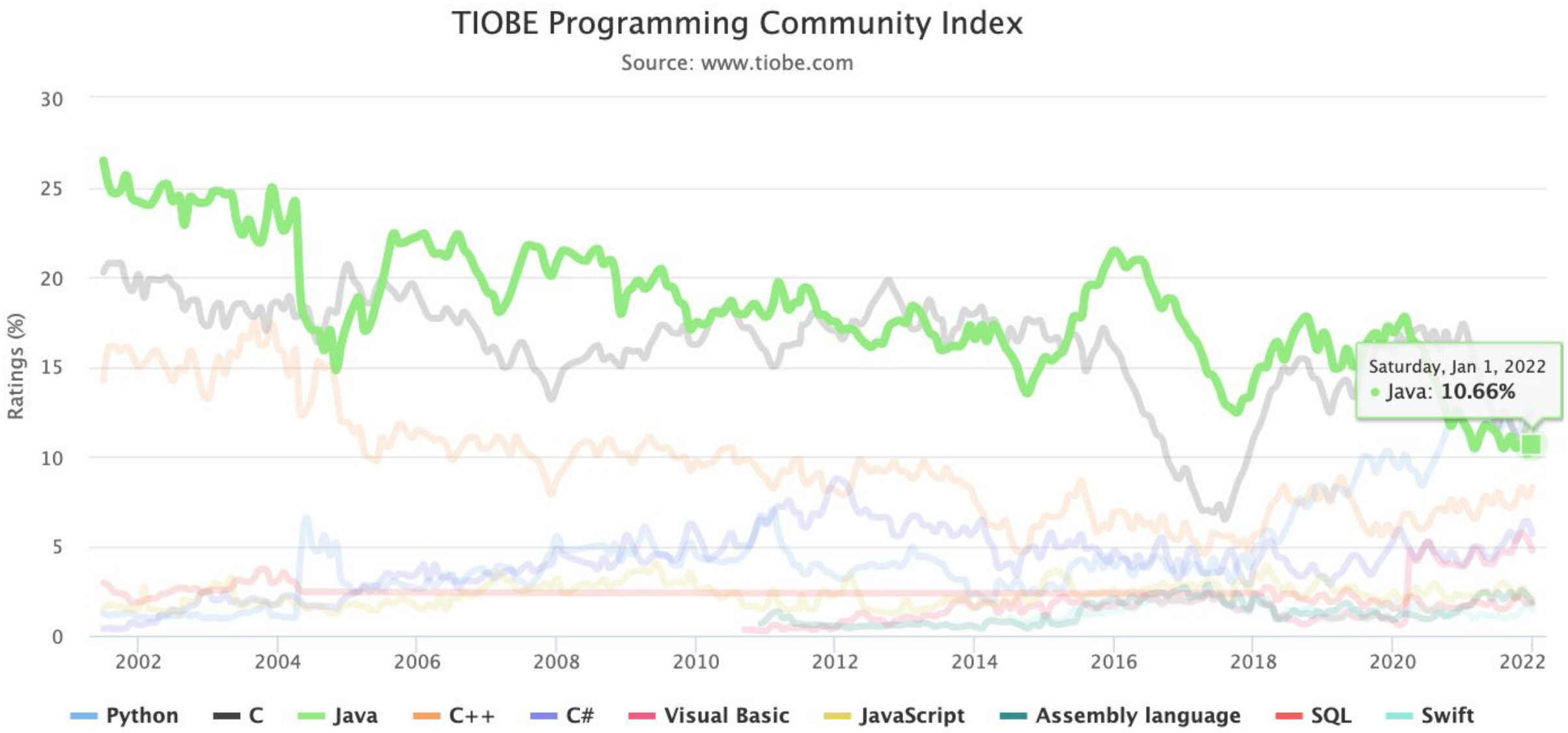
of companies use Java to build over 80% of their applications

**#3 Stability**

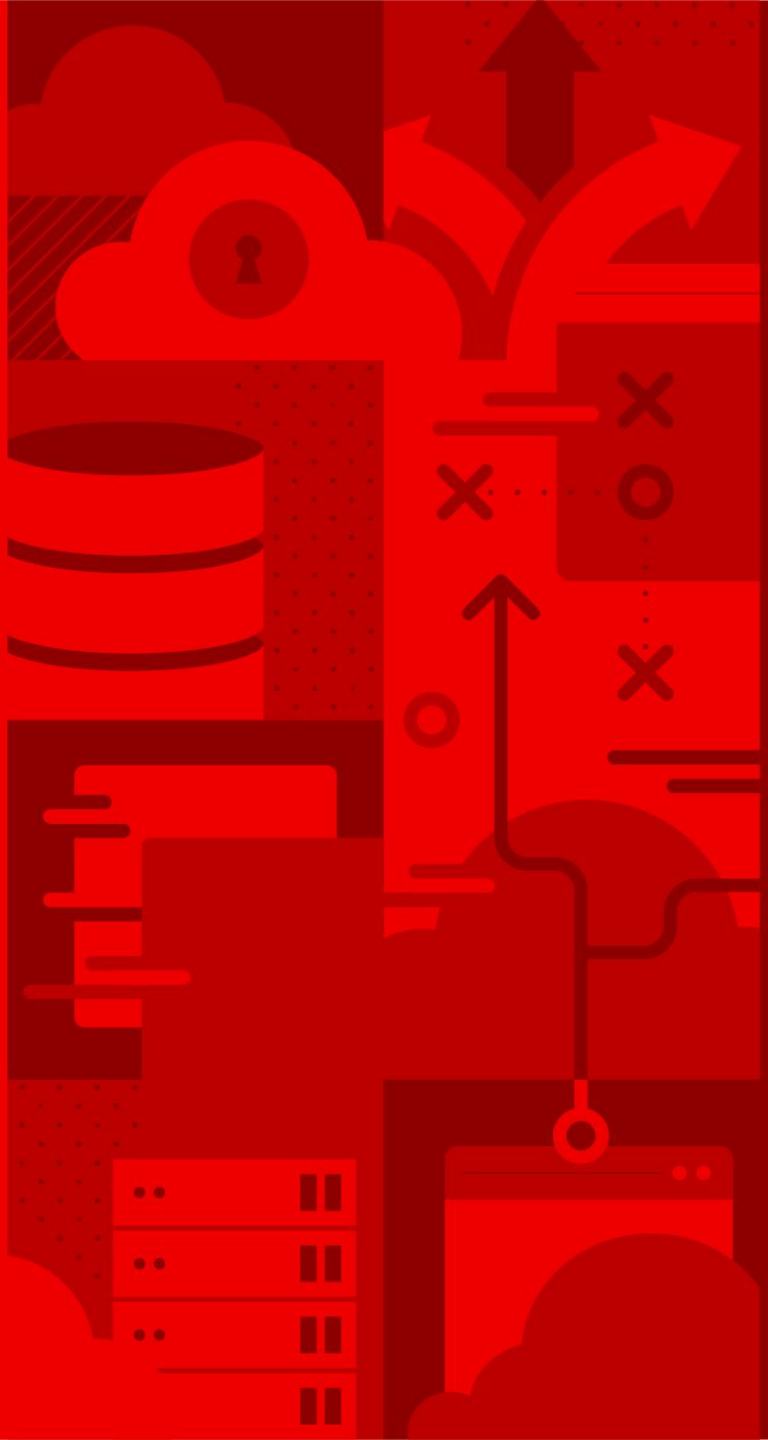
Red Hat


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
Python
C
Java
C++
C#
Visual Basic
JavaScript
Assembly language
SQL
Swift

Red Hat


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
Saturday, Jan 1, 2022
Java: 10.66%
Python
C
Java
C++
C#
Visual Basic
JavaScript
Assembly language
SQL
Swift

Red Hat

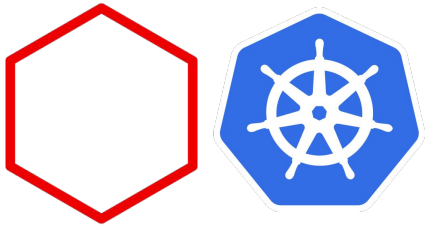


# New architectures drives new technology needs

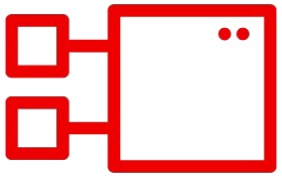
Red Hat

# Technology trends

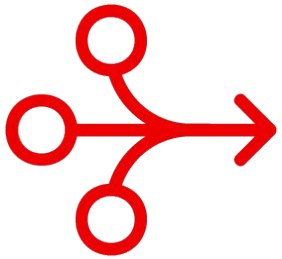
Cloud and Edge Computing



Containers and Kubernetes



Microservices Architecture



Event-driven Architectures and reactive systems

Serverless and FaaS

# *"Historical"* Enterprise Java Stack

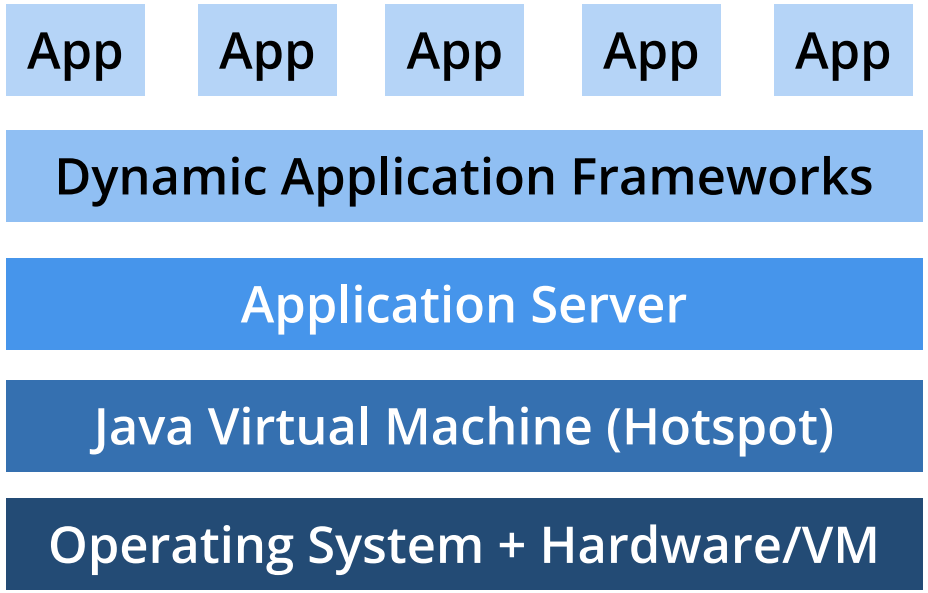
Architecture: Monoliths

Deployment: multi-app, appserver

App Lifecycle: Months

Memory: 1GB+ RAM

Startup Time: 10s of sec



# *"Modern"* Enterprise Java Stack

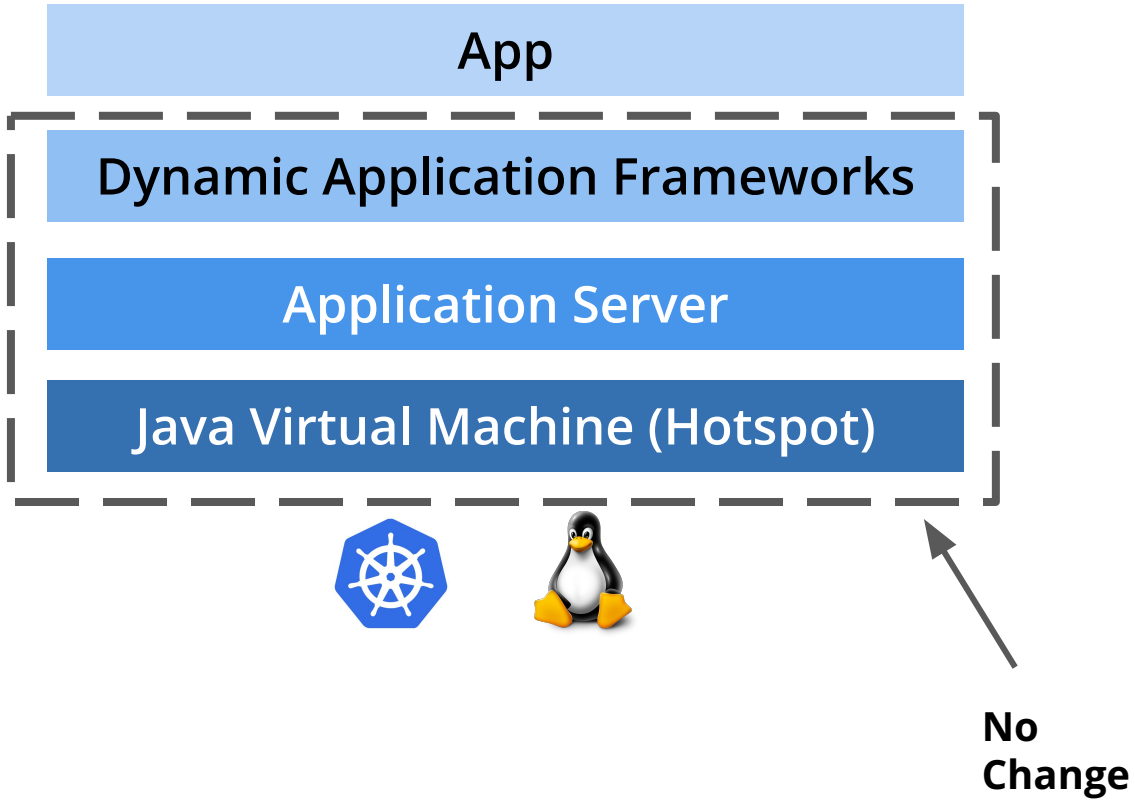
Architecture: Microservices

Deployment: Single App

App Lifecycle: Days

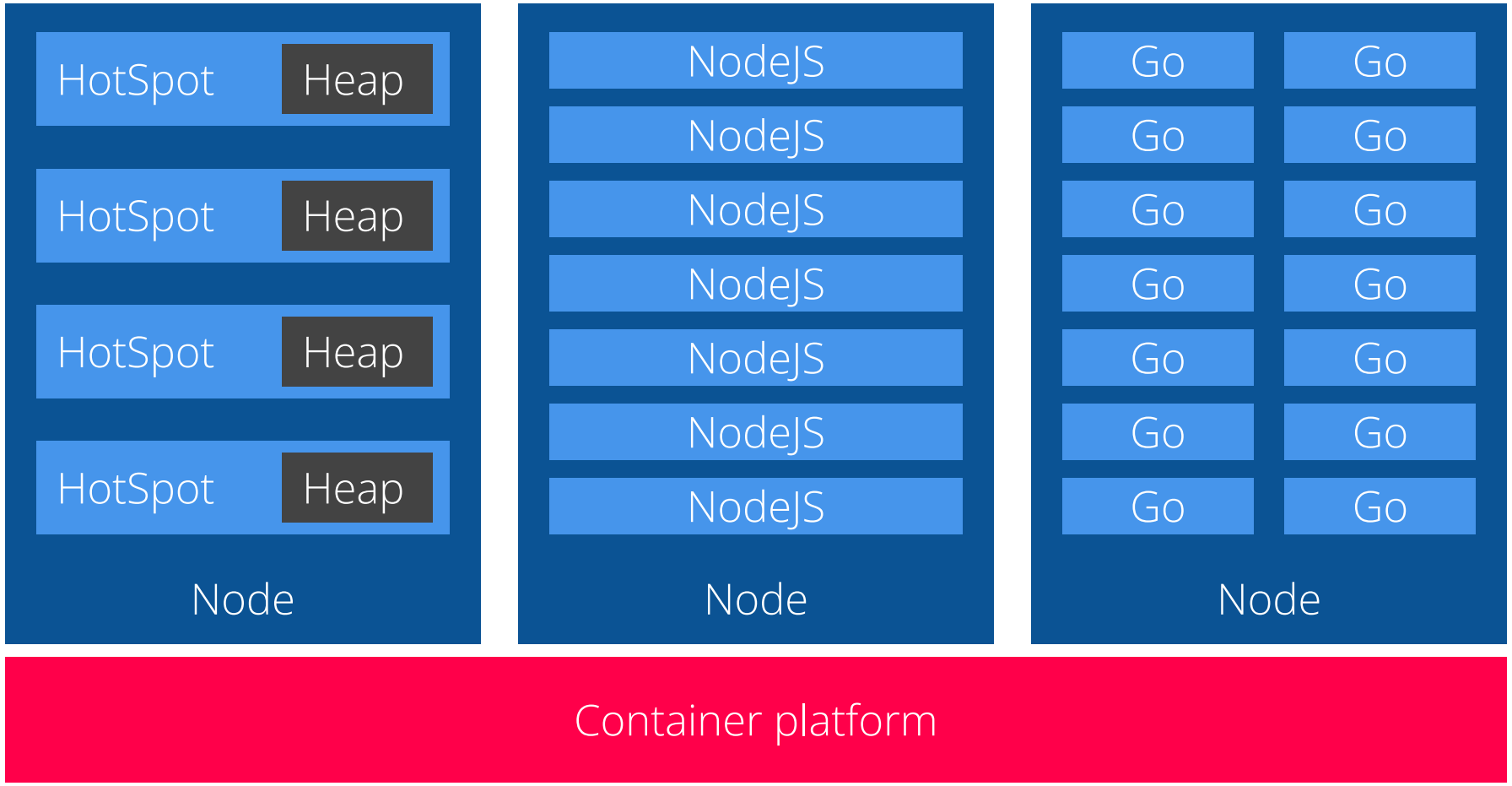
Memory: 100MBs+ RAM

Startup Time: Seconds




Red Hat

# Hidden Truth About Java + Containers



# THERE IS A NEED FOR A NEW JAVA STACK FOR CLOUD-NATIVE AND SERVERLESS

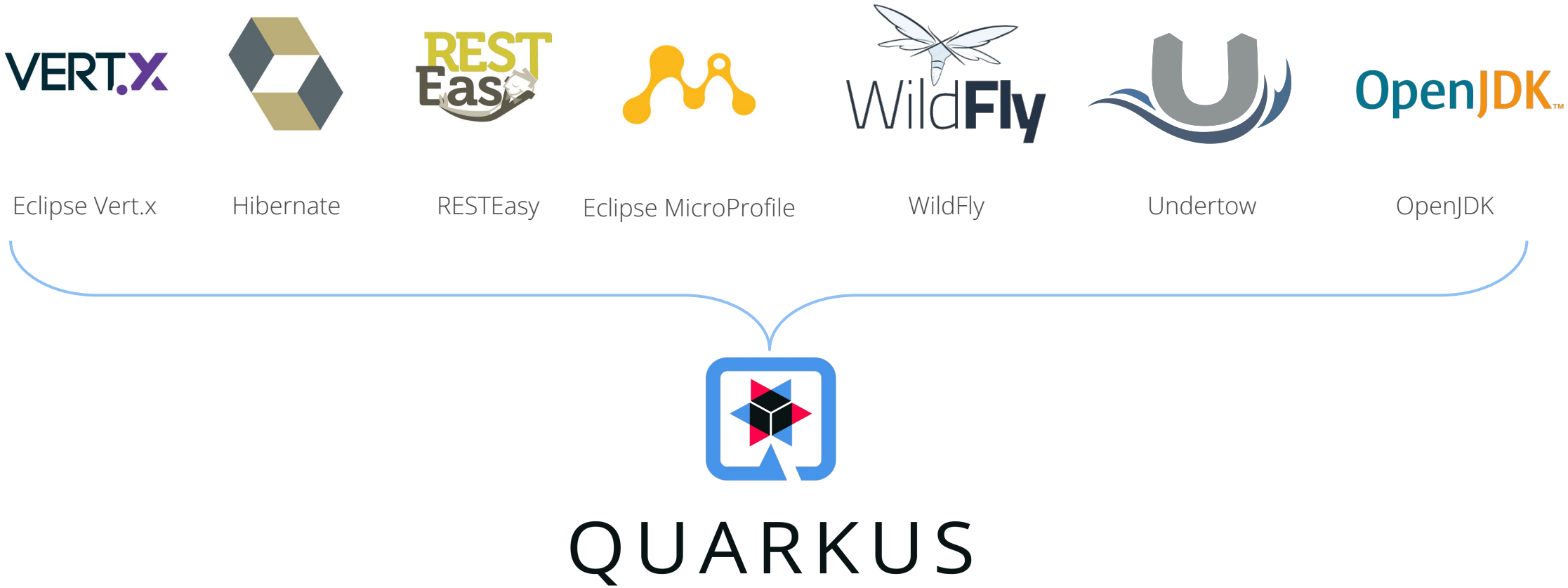
Red Hat

# QUARKUS

Supersonic. Subatomic. Java.

Red Hat

# Experts from cloud-native Java OS projects



# Benefits

## Container First

Tailors your app for HotSpot & GraalVM

Fast boot time and low RSS memory

Serverless fit

## Developer Joy

Live coding

Unified configuration

Frictionless local dev with dev services

## Unifies Imperative & Reactive

Combines blocking and non-blocking

Built-in event bus

## Best of Breed Libraries & Standards

500+ extensions

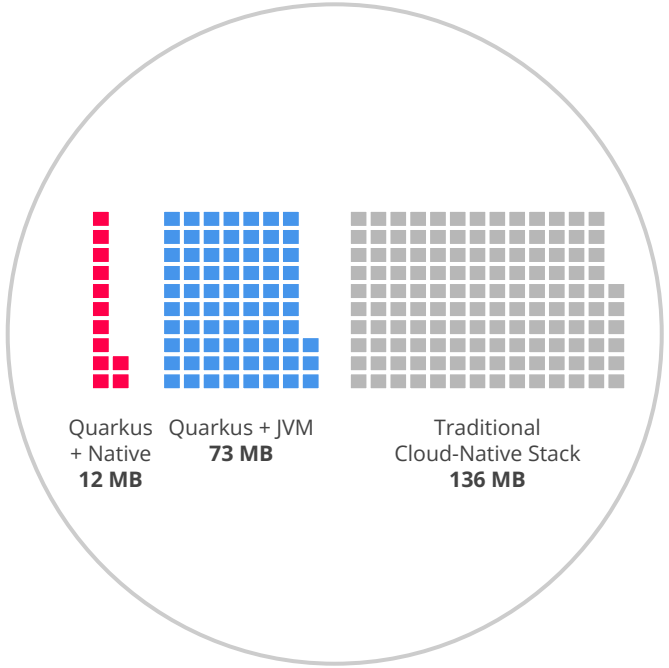
“Powered by Quarkus” applications

Red Hat

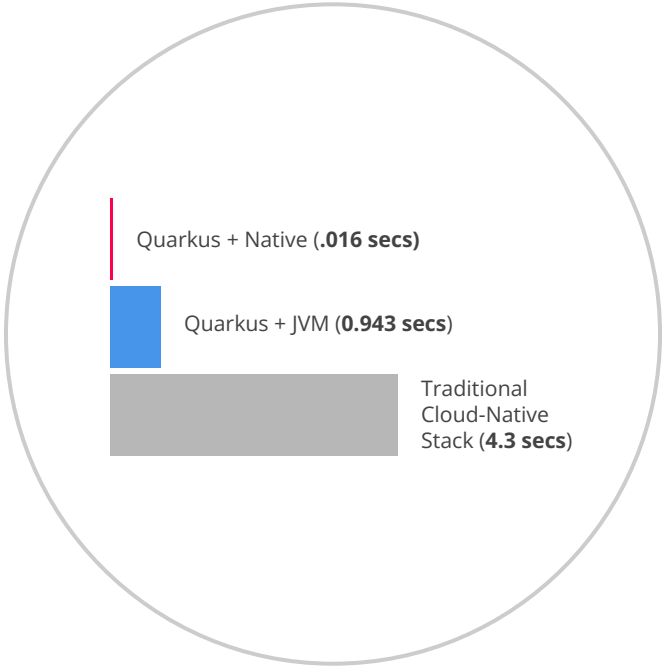
# Benefit No. 1: Container First

*"We went from **1-min** startup times to **400 milliseconds**"*

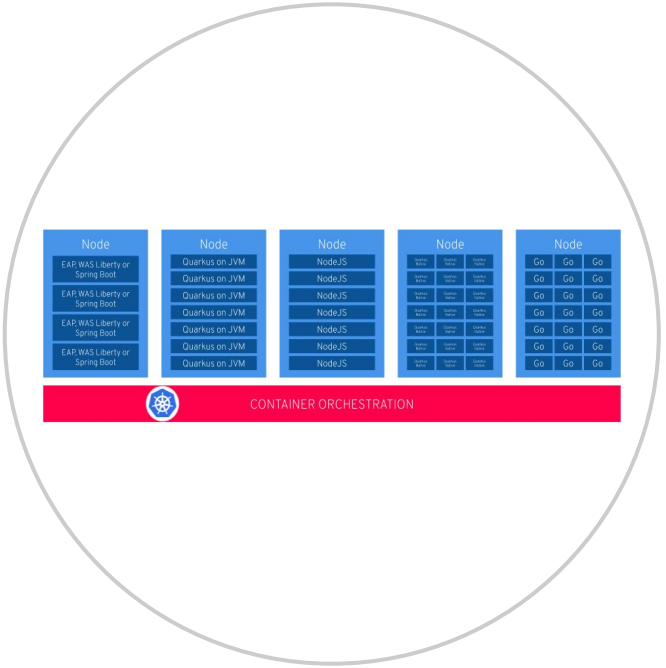
## *Reduced Memory Footprint*




## *Fast Startup Time*

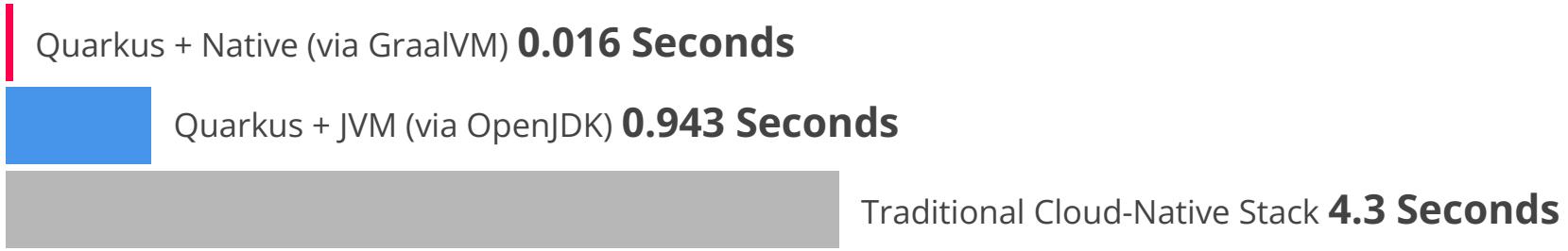



## *Smaller Disk Footprint*

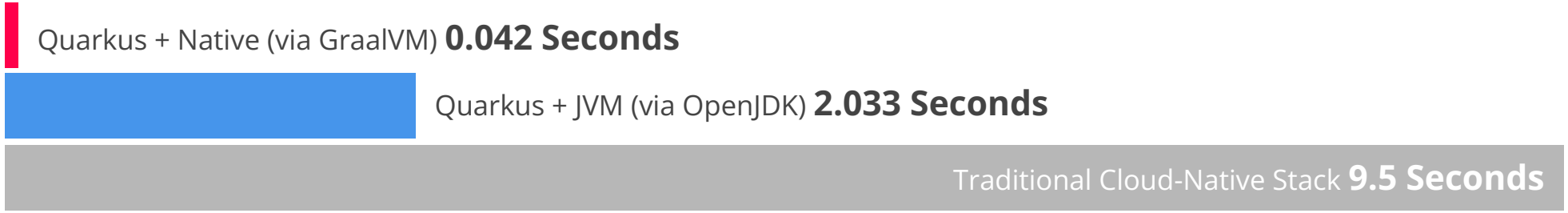


# Supersonic, Subatomic Java

## REST




## REST + CRUD

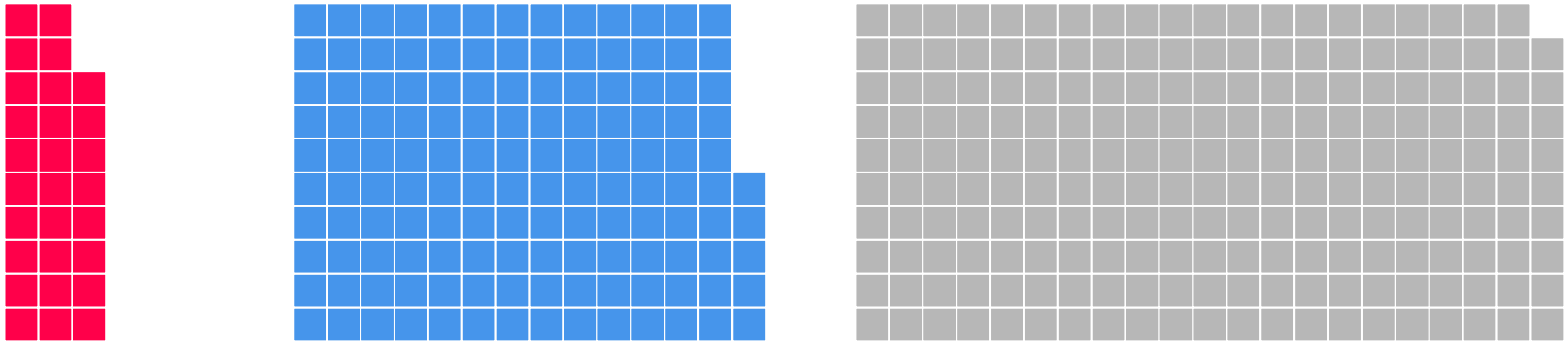



Time to first response

Red Hat

# Supersonic, Subatomic Java

REST + CRUD*



Quarkus + Native
(via GraalVM)
**28 MB**

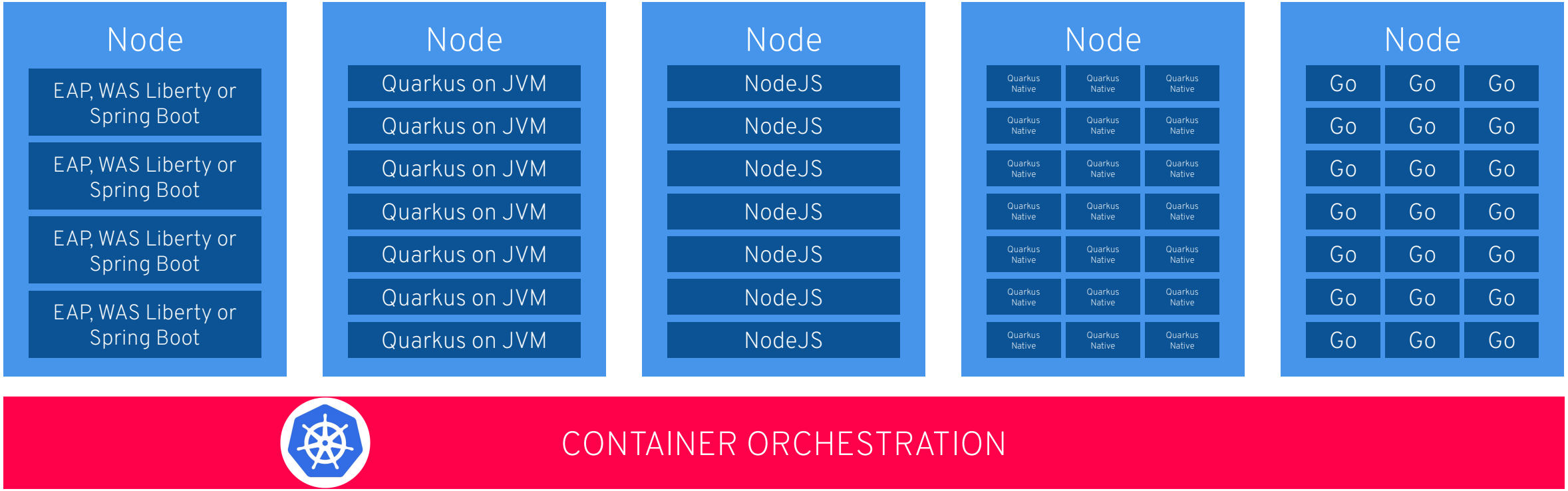
Quarkus + JVM
(via OpenJDK)
**145 MB**

Traditional
Cloud-Native Stack
**209 MB**

*Memory (RSS) in Megabytes, tested on a single-core machine

Red Hat

# Cloud Native Java Stack + Containers



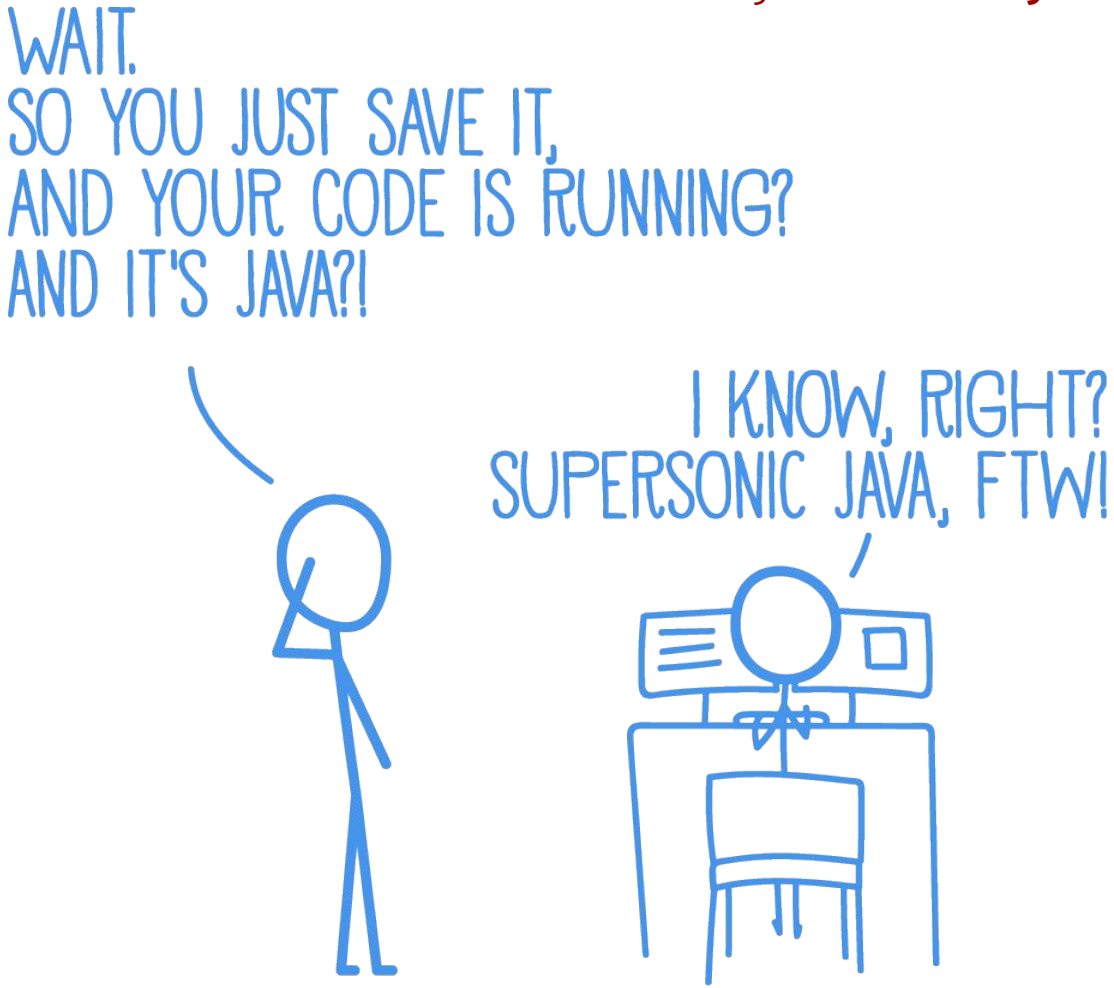

*"We could run **3 times** denser deployments without sacrificing **availability** and **response times** of services"*

https://developers.redhat.com/blog/2017/03/14/java-inside-docker/

Red Hat

# Benefit No. 2: Developer Joy

*"Our developers used to wait **2 to 3 mins** to see their changes. **Live coding** does away with this."*

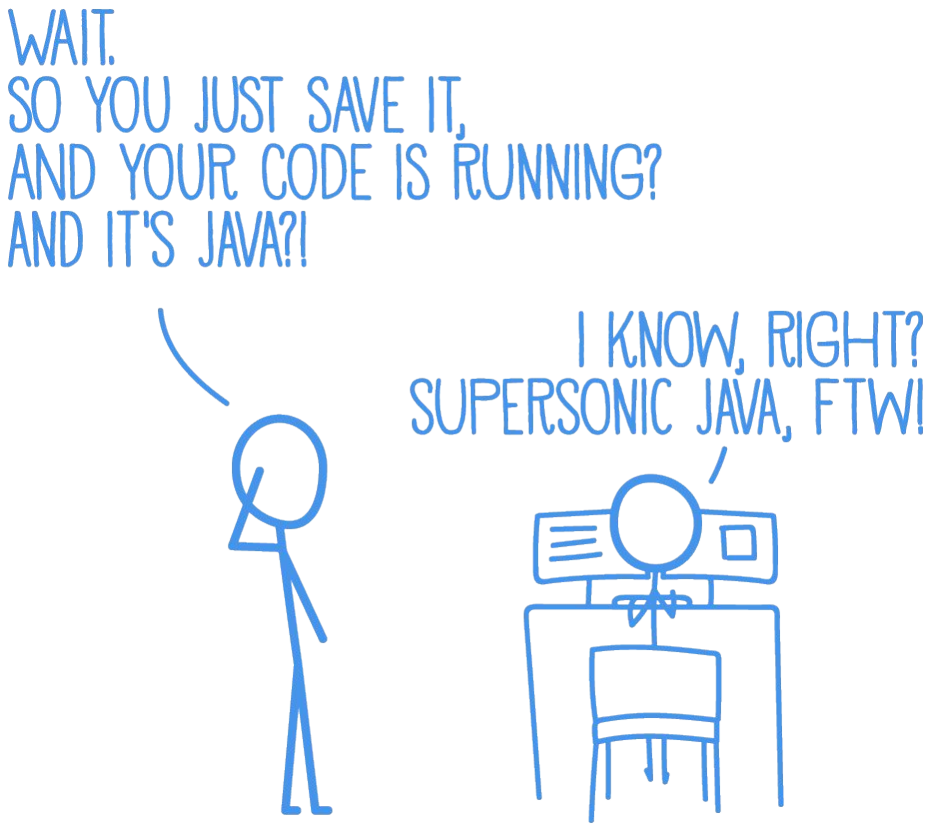


# Benefit No. 2: Developer Joy

*"Our developers used to wait **2 to 3 mins** to see their changes. **Live coding** does away with this."*

## A cohesive platform for optimized developer joy:

- Based on standards and more
- Unified configuration
- Live coding
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation
- Zero configuration with dev services
- Continuous testing for instant feedback




Red Hat

# Benefit No. 3: Unifies Imperative and Reactive

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
     return say.hello();
}
```

```
@Inject @Stream("kafka")
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Publisher<String> stream() {
     return reactiveSay;
}
```

Red Hat

# Benefit No. 3: Unifies Imperative and Reactive

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
     return say.hello();
}
```

```
@Inject @Stream("kafka")
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Publisher<String> stream() {
     return reactiveSay;
}
```

- Combine both Reactive and imperative development in the same application
- Inject the EventBus or the Vertx context
- Use the technology that fits your use-case
- Key for reactive systems based on event driven apps

Red Hat

# Benefit No. 4: Best of Breed Frameworks & Standards

*"When you adopt Quarkus, you will be productive from day one since you don't need to learn new technologies."*

Eclipse Vert.x

Hibernate

RESTEasy

Apache Camel

Eclipse MicroProfile

Netty

Kubernetes

OpenShift

Jaeger

Prometheus

Apache Kafka

Infinispan

Flyway

Neo4j

MongoDB

MQTT

KeyCloak

Apache Tika

# Use Cases

## NET NEW

Low memory footprint + lightning fast startup time + small disk footprint = an ideal runtime for Kubernetes-native microservices

## MONO 2 MICRO

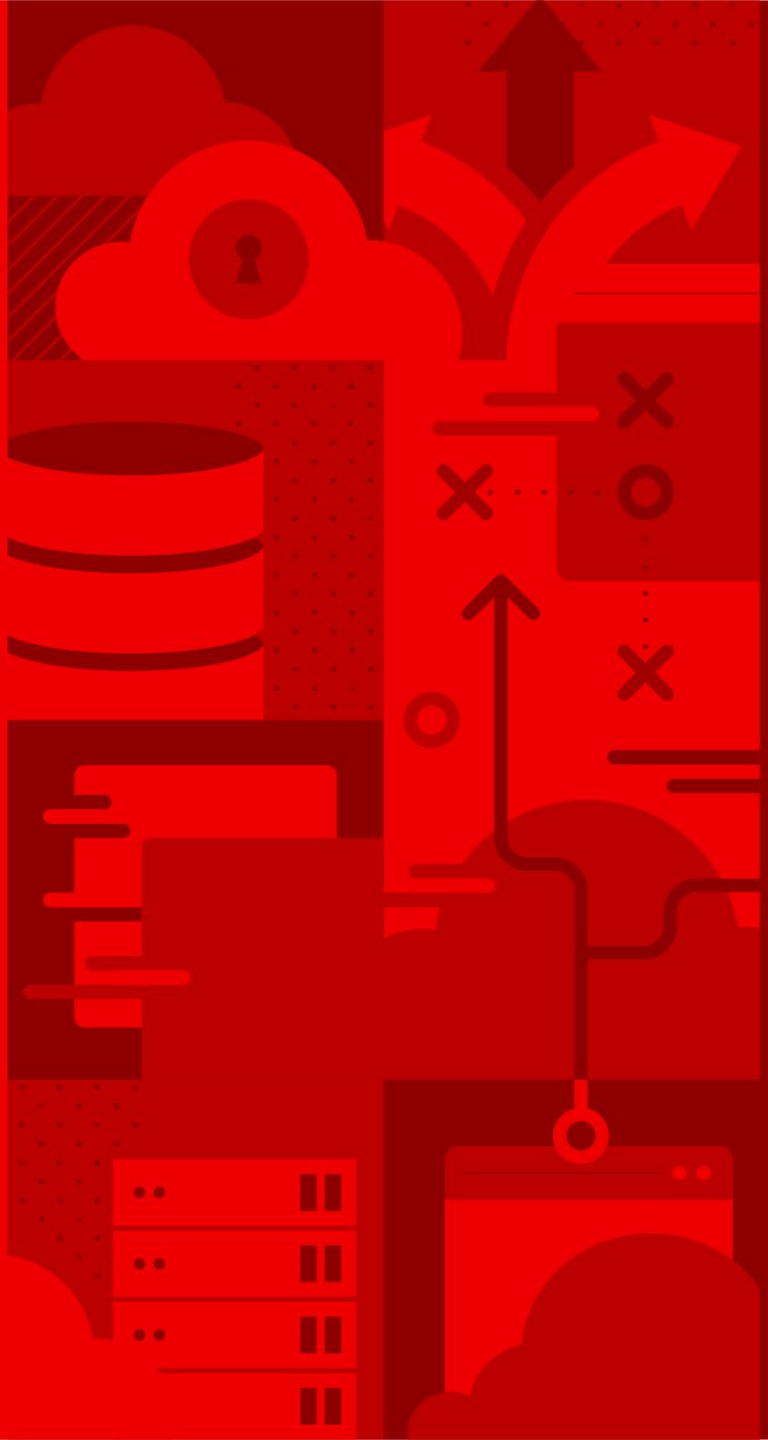
Quarkus is a great choice to modernize existing monolithic applications by breaking it into  smaller, loosely coupled microservices.

## SERVERLESS

Scaling up or down (0) is extremely fast with Quarkus making it an ideal runtime for creating serverless applications.

## EVENT-DRIVEN/REACTIVE

Quarkus utilizes an asynchronous, reactive event loop that makes it easy to create reactive applications.



# Demo

Red Hat

# HOW DOES QUARKUS WORK?

Red Hat

# Quarkus - Optimizing the Java Stack

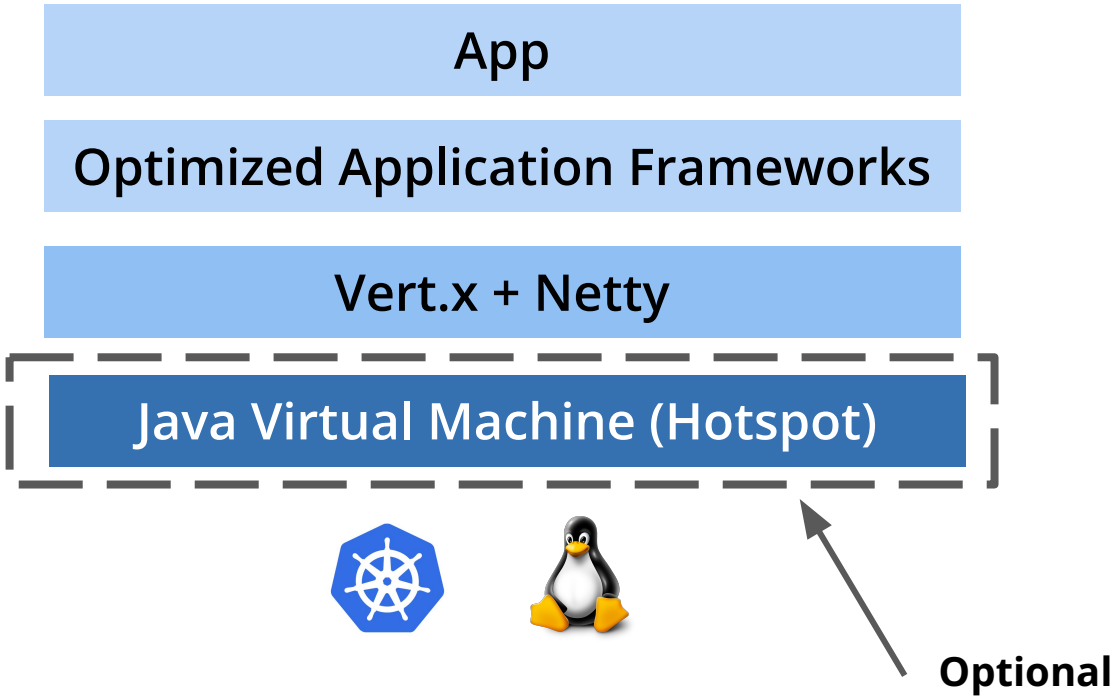
Architecture: Microservices, Serverless

Deployment: Single App

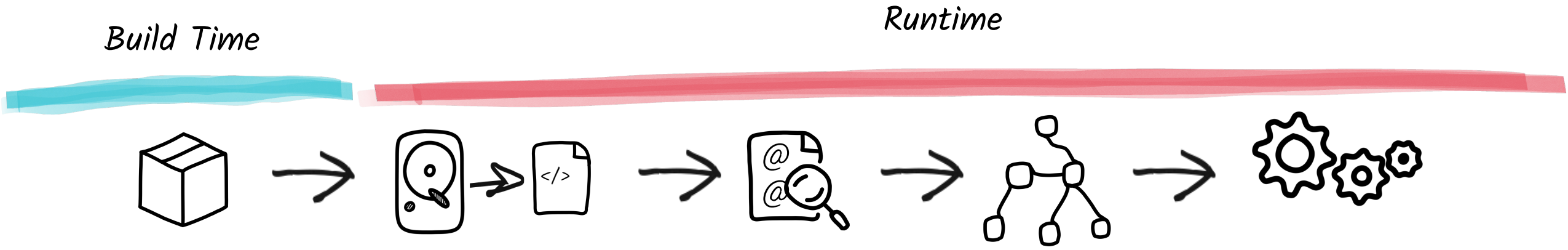
App Lifecycle: Milliseconds to Days

Memory: 10MBs+ RAM

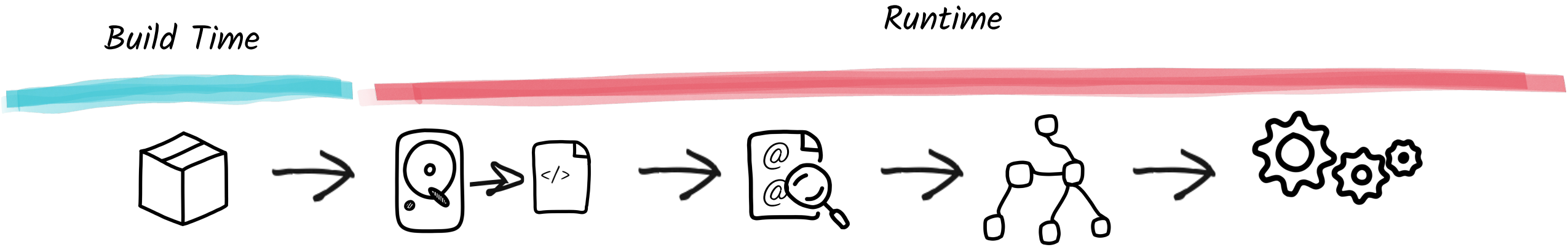
Startup Time: Milliseconds



# How Does a Framework Start?



# How Does a Framework Start?




Packaging (maven, gradle...)

Load config file from file system Parse it

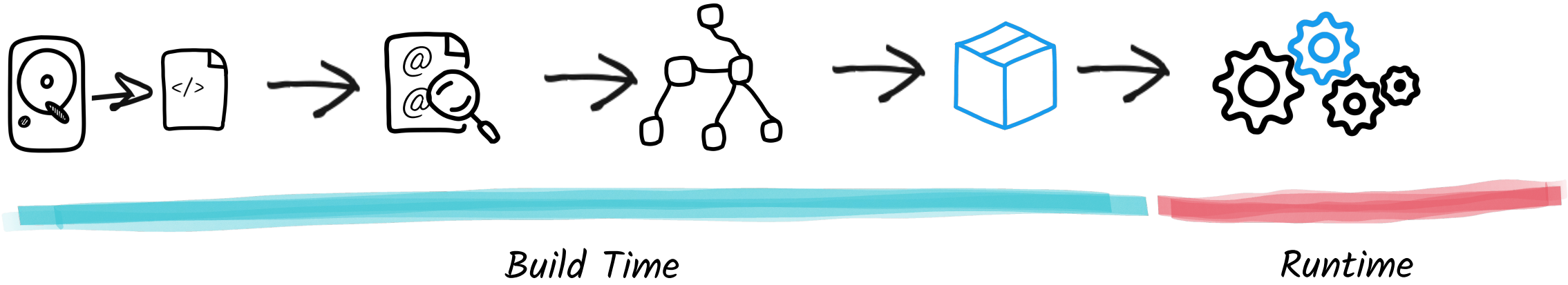
Classpath scanning to find annotated classes Attempt to load class to enable/disable features

Build its model of the world.

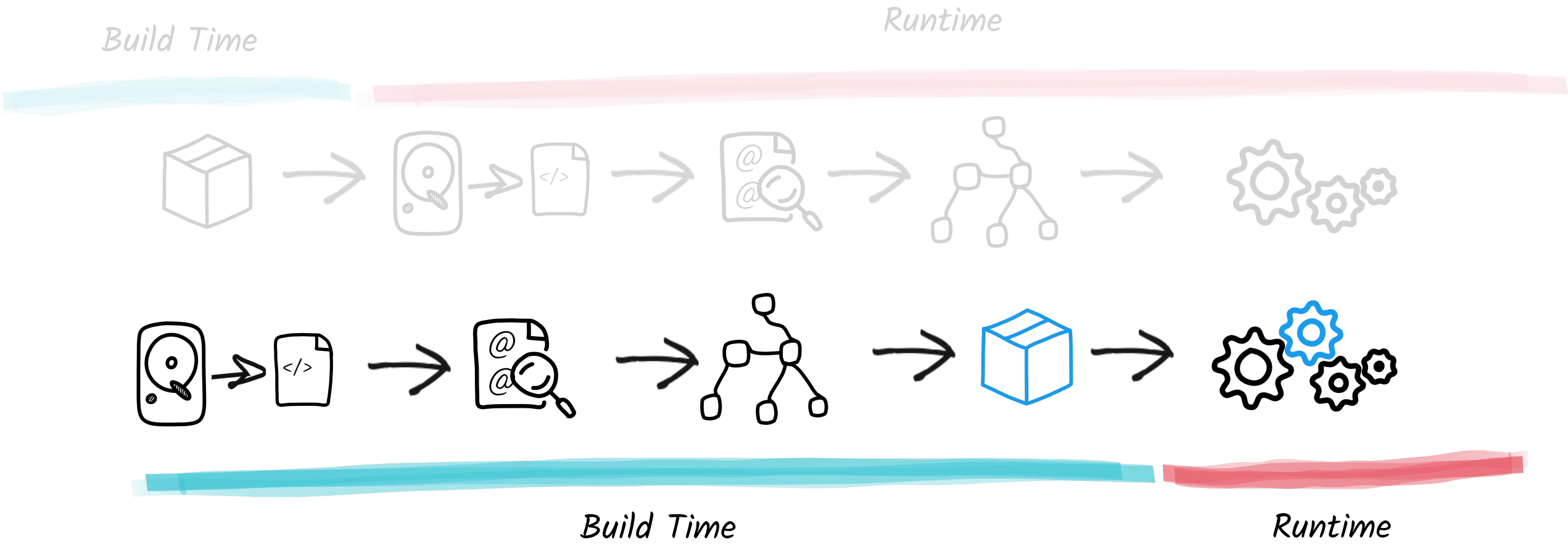
Start the management (thread, pool...)

Red Hat

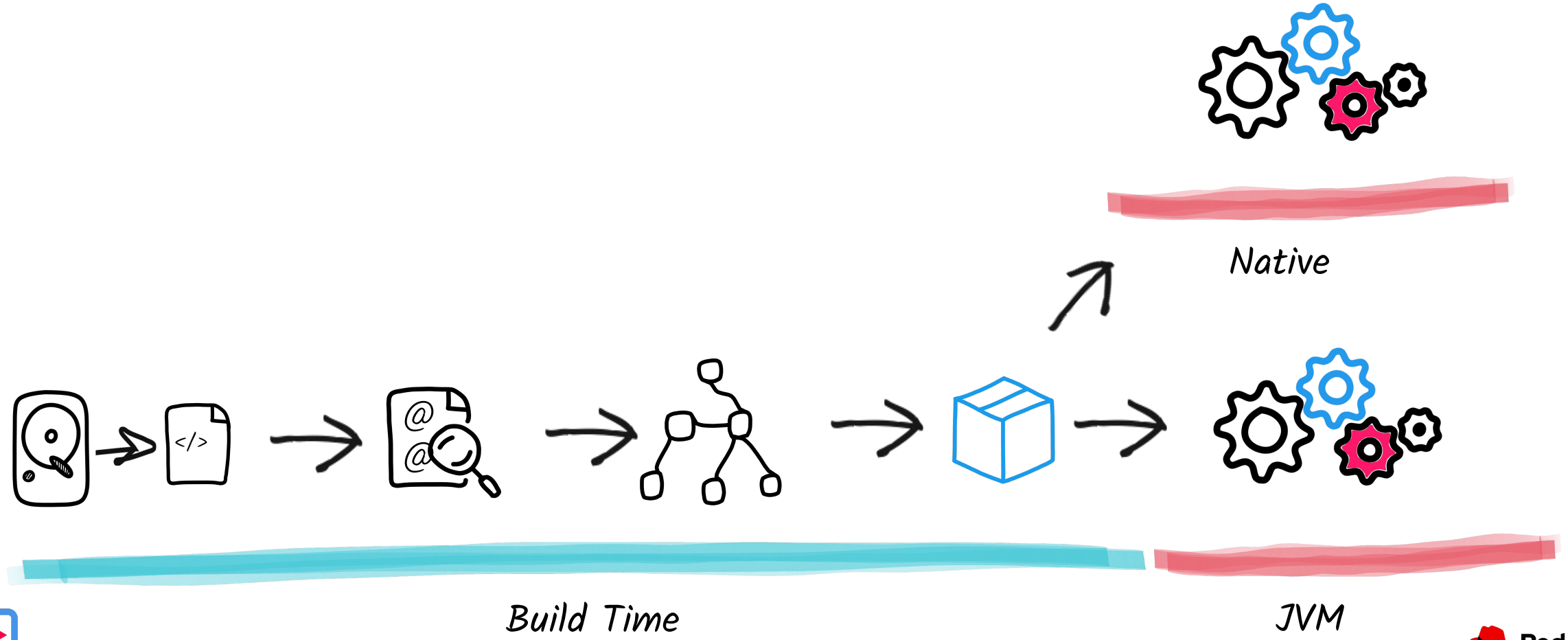
# The Quarkus Way



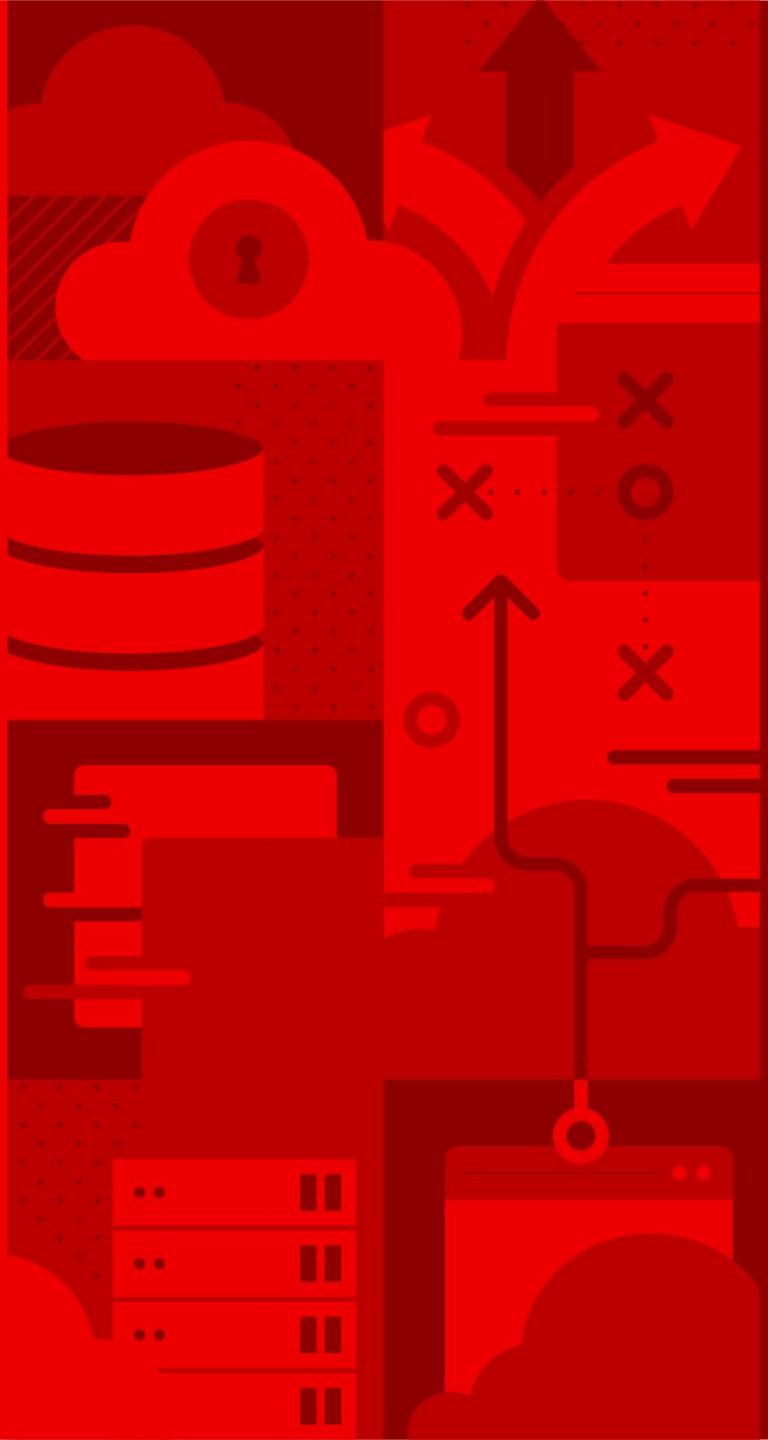
# The Quarkus Way



# The Quarkus Way enables Native compilation




Red Hat



# Case studies

Red Hat

# Customers using Quarkus Today

Lufthansa

"We could run 3 times denser deployments without sacrificing availability and response times of service"

**Thorsten Pohl**
Lufthansa Technik AVIATAR
Product Owner Automation &
Platform Architect

talkdesk

"When you adopt Quarkus, you will be productive from day one since you don't really need to learn new technologies."

**Roberto Cortez**
Talkdesk Principal Architect

vodafone

"Quarkus seemed to provide the performance boost we needed while at the same time having a good backer (Red Hat) and relying on battle-tested technologies"

**Christos Sotiriou**
DXL technical lead at Vodafone Greece

vodafone

"Quarkus seemed to provide the performance boost we needed while at the same time having a good backer (Red Hat) and relying on battle-tested technologies"

**Christos Sotiriou**
DXL technical lead at Vodafone Greece

## Challenge

Running 140 microservices, with heavy spikes in traffic, caused delays and pause while booting new containerized applications leading to waste of marketing efforts.

## Solution

After initial tests indicated that Quarkus would reduce application boot times, reduce CPU and memory usage, and make the entire development process run faster, Vodafone decided to port their most essential libraries and microservices to this new stack.

## Why Quarkus

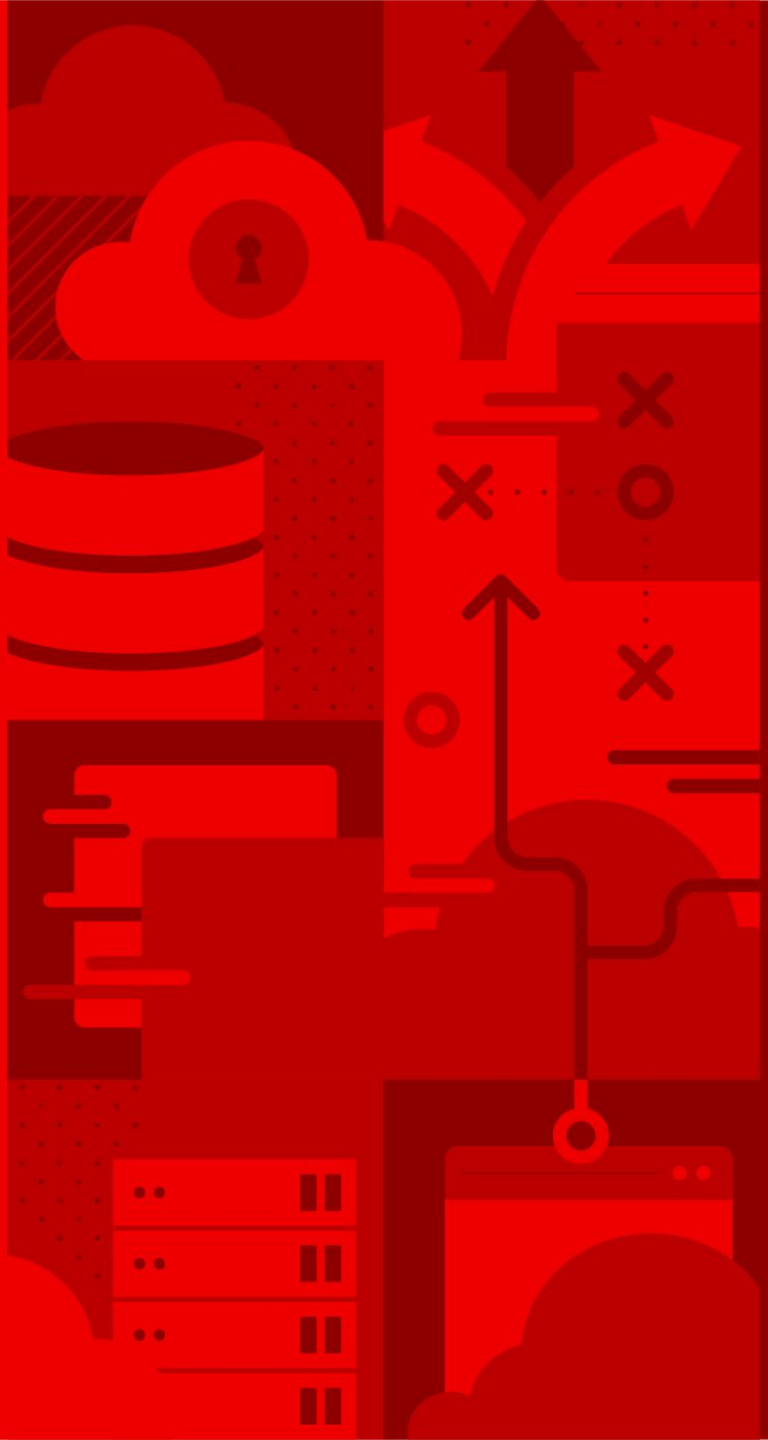
The main criteria for their selection process to find a replacement for Spring Boot were developer efficiency, lower cloud resource consumption and shorter applications boot-up times. A great impact on cloud resource consumption costs as well as user experience improvement. Their trust of Red Hat combined with its credibility in the software market gave them the assurance that they were making the right choice by selecting Quarkus, whose sponsor is Red Hat.

## Results

- Start-up times have been reduced to almost a quarter without any optimization
- Memory resource consumption was cut in half in JVM mode
- The use of the Quarkus live coding capability (a.k.a. dev mode) resulted in an increase of developer productivity
- Migrating from Spring Boot to Quarkus didn't require a lot of effort for their Spring developers, resulting in a small learning curve
- Far healthier cluster overall, as it is no longer experiencing difficulty in handling the sudden traffic spikes driven by the company's direct marketing campaigns

Source: Vodafone Greece replaces Spring Boot with Quarkus, Nov. 2019.

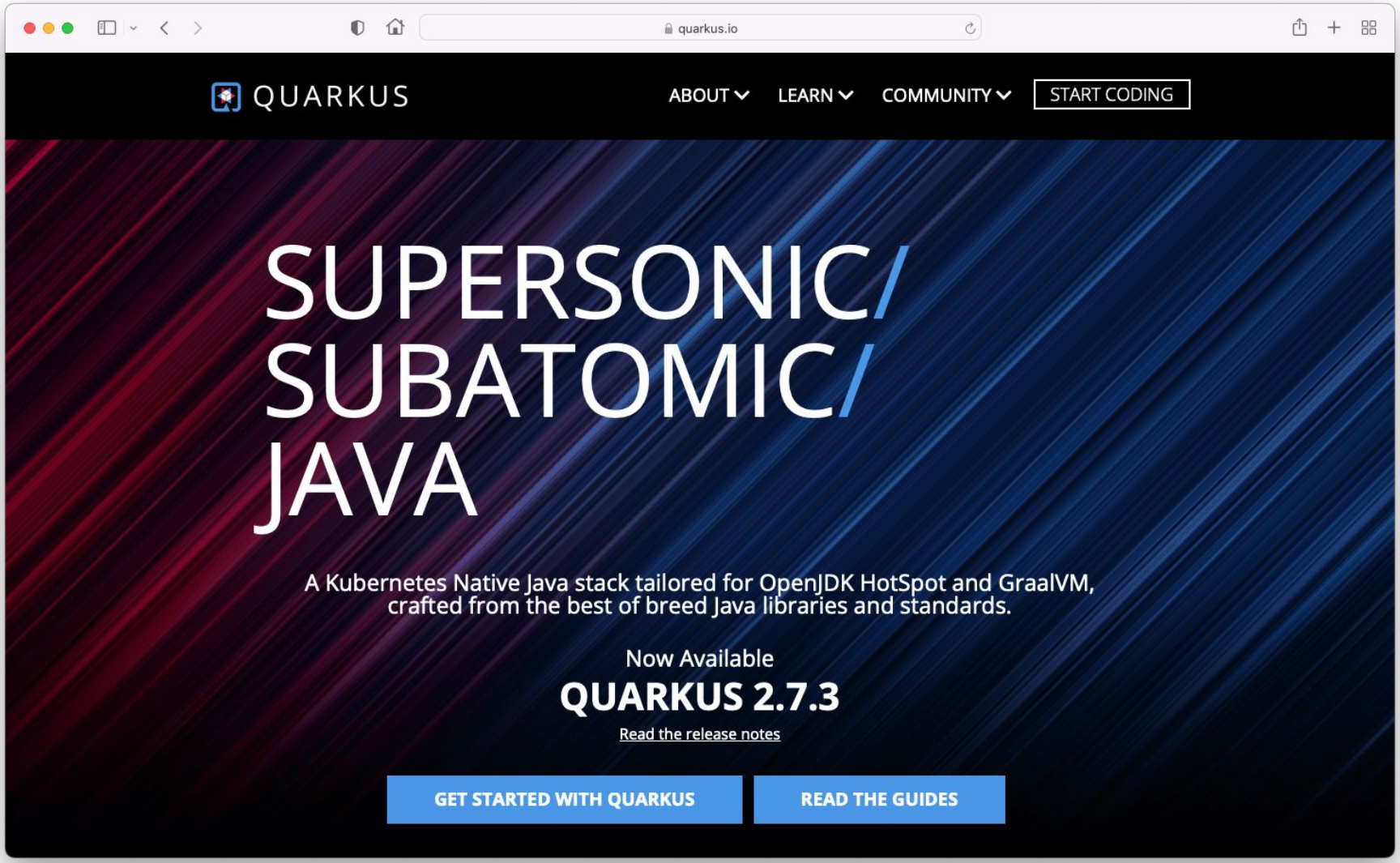
Red Hat



# Where to learn more?

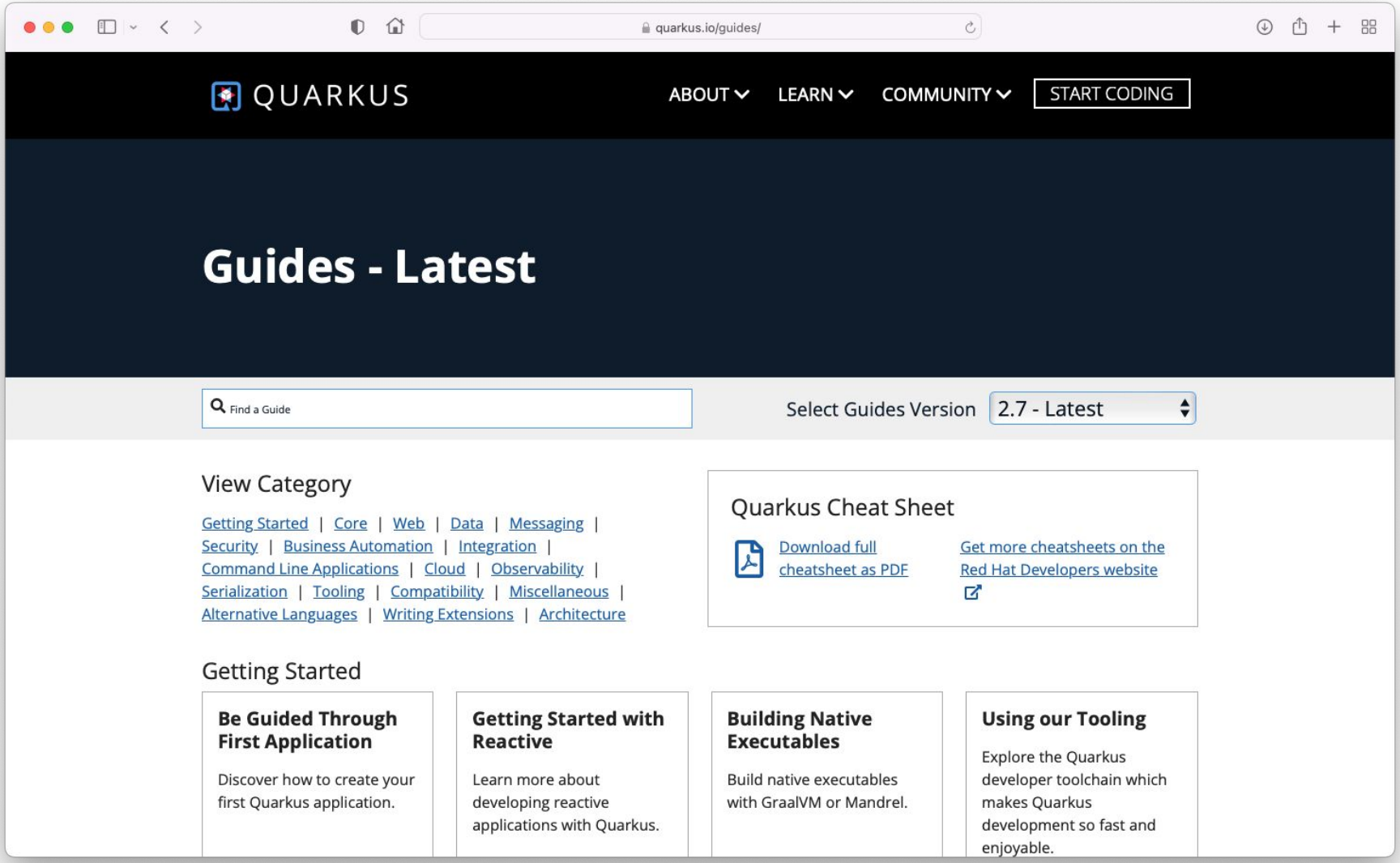
Red Hat

# quarkus.io




Red Hat

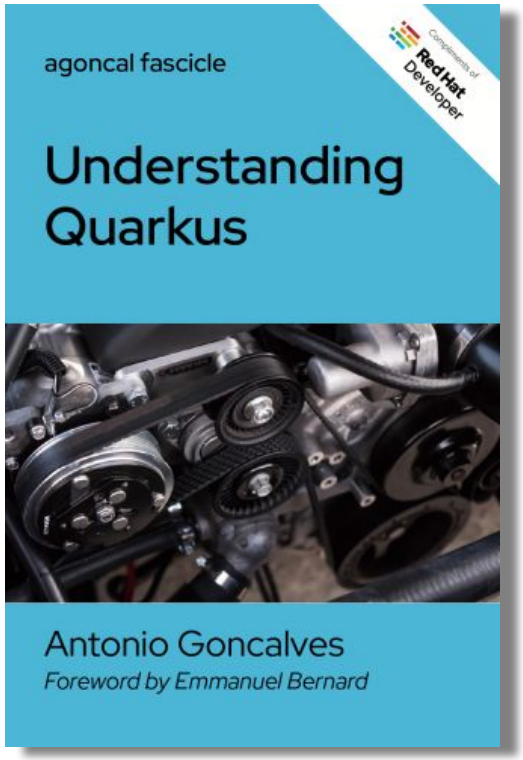
# quarkus.io/guides/

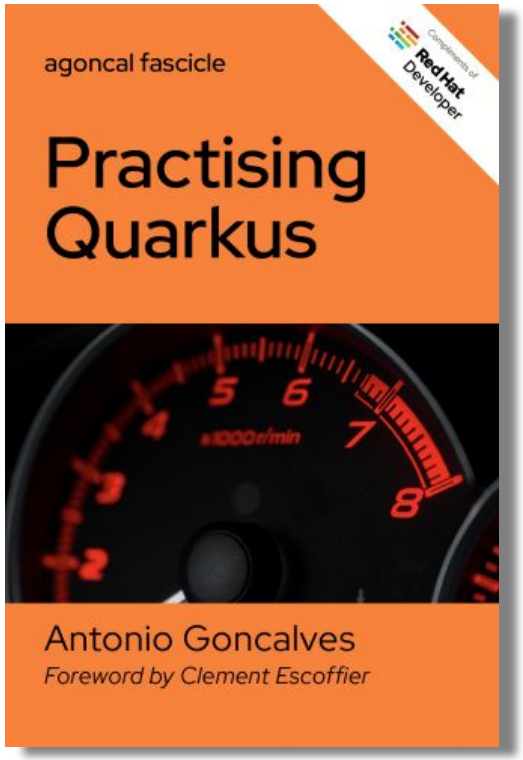


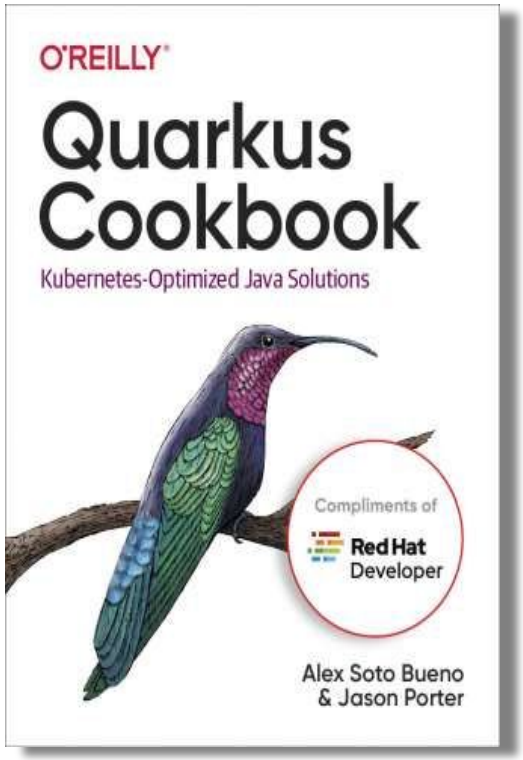

Red Hat

free e-books

# https://developers.redhat.com/e-books/





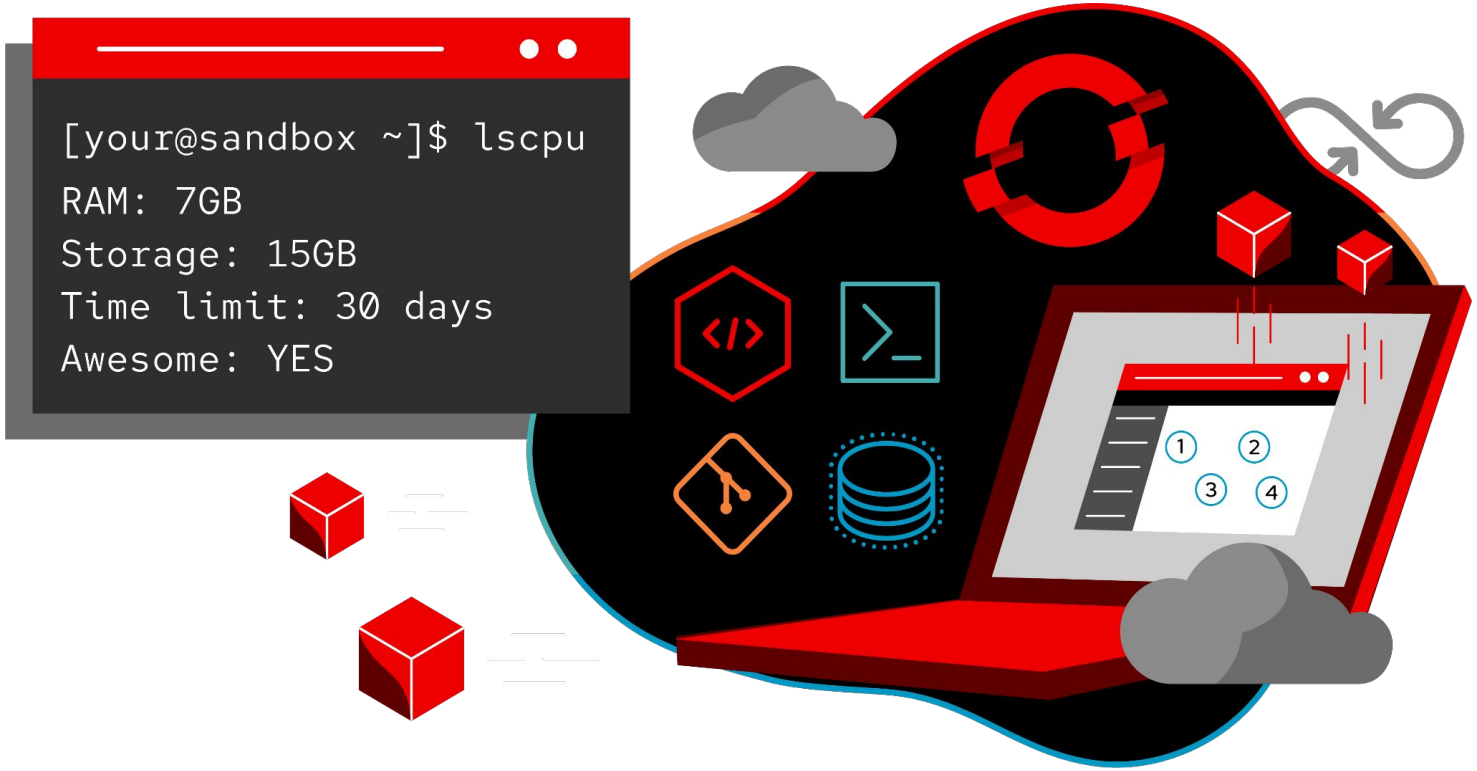
Red Hat

Red Hat Developer

# Developer Sandbox

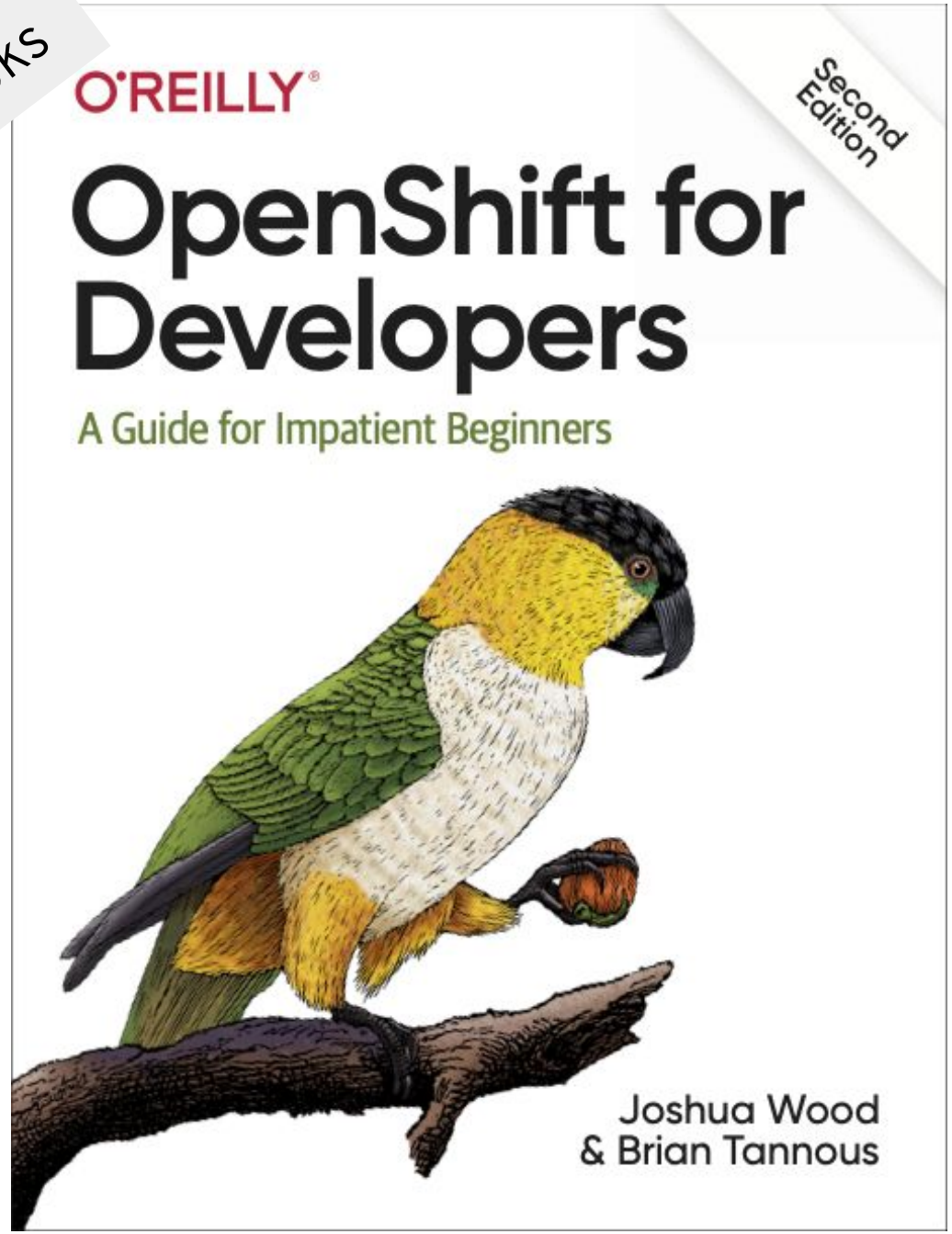
Get **free access** for renewable **30 days** to a self-service, cloud-hosted **Kubernetes** experience with **Developer Sandbox** for **Red Hat OpenShift**.

**https://developers.redhat.com/developer-sandbox**




Red Hat

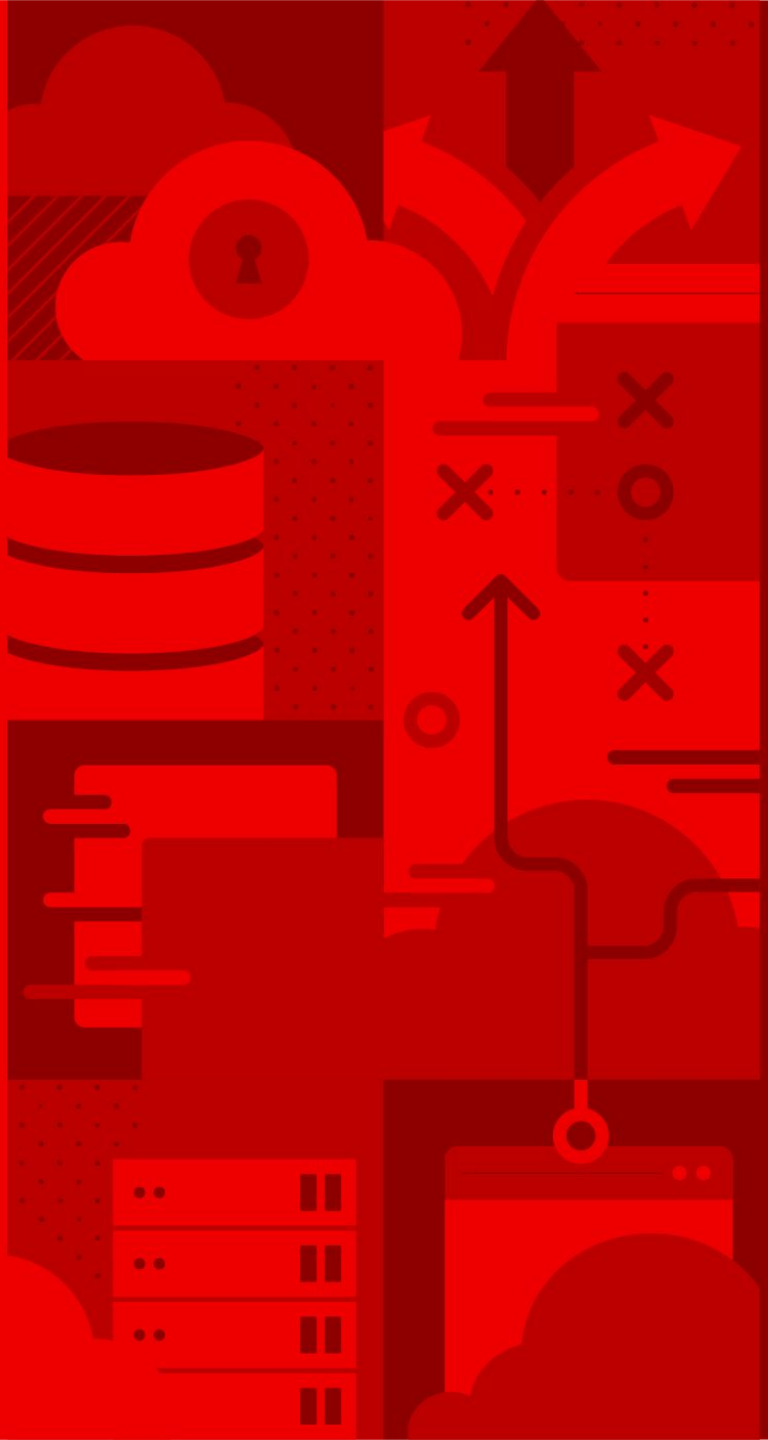
free e-books




[Download](https://red.ht/3lxJCzY)

https://red.ht/3lxJCzY

Red Hat



# Summary

Red Hat

# Summary

- New architectures and design principles drive new needs on technology. New requirements on Java to stay relevant.

  e.g. Cloud and Edge Computing, Containers & K8S, MSA, EDA, Serverless/FaaS

- Red Hat is investing in upstream projects that modernise Java to meet new needs.

QUARKUS

- Quarkus has superfast startup times and low memory consumption, and at the same time provide a very pleasant and productive experience for developers.

- Helps organisations to protect Java investments & skill sets to modernise legacy as well as develop the next generation of cloud native applications.

Red Hat

# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

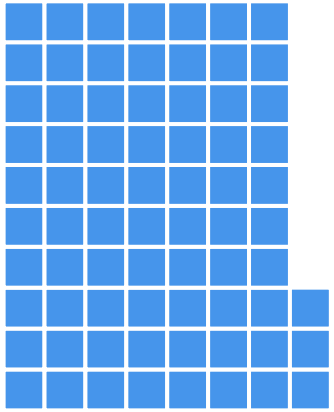


Red Hat

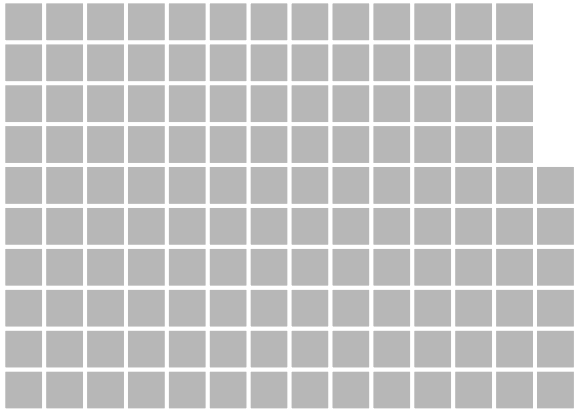
# Supersonic, Subatomic Java

REST*

Quarkus + Native
(via GraalVM)
**12 MB**



Quarkus + JVM
(via OpenJDK)
**73 MB**



Traditional
Cloud-Native Stack
**136 MB**

*Memory (RSS) in Megabytes, tested on a single-core machine

Red Hat

# Quarkus Tools - Build


Quarkus Application Configurator

QUARKUS CLI

```
quarkus create app bar
```

maven

Gradle

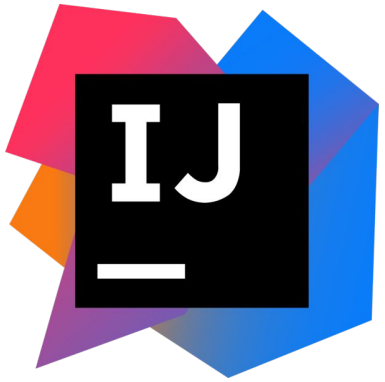
```
mvn io.quarkus:quarkus-maven-plugin:2.7.3.Final-redhat-00001:create
\
    -DprojectGroupId=org.acme \
    -DprojectArtifactId=getting-started \
    -DplatformGroupId=com.redhat.quarkus \
    -DplatformVersion=1.3.2.Final-redhat-00001 \
    -DclassName="org.acme.quickstart.GreetingResource" \
    -Dpath="/hello"
cd getting-started
```

Red Hat

# Quarkus Tools - IDE

VSCode

Eclipse




IntelliJ

che.openshift.io

Red Hat

# Supersonic, Subatomic Java with Quarkus


Quarkus provides an effective solution for running Java applications that deal in microservices, containers, Kubernetes, serverless, or cloud in general.

- Optimized Java framework with low memory consumption and blazingly fast first response times.
- Allows developers to get their job done faster with a low learning curve.
- Unifies imperative and reactive programming models
- Compatible with popular frameworks like Eclipse MicroProfile, select Spring APIs, Hibernate, and more

# Kubernetes-Native Java from First Principles

## Polyglot - power and responsibility

The power of choosing any language needs to be tempered with choosing the right language. Language features are only valuable if you have developer experience in that language.

## Toolchain beyond the desktop

Cloud-native application development extends beyond the IDE, introducing unique challenges for inner/outer loop development and CI/CD pipeline automation.

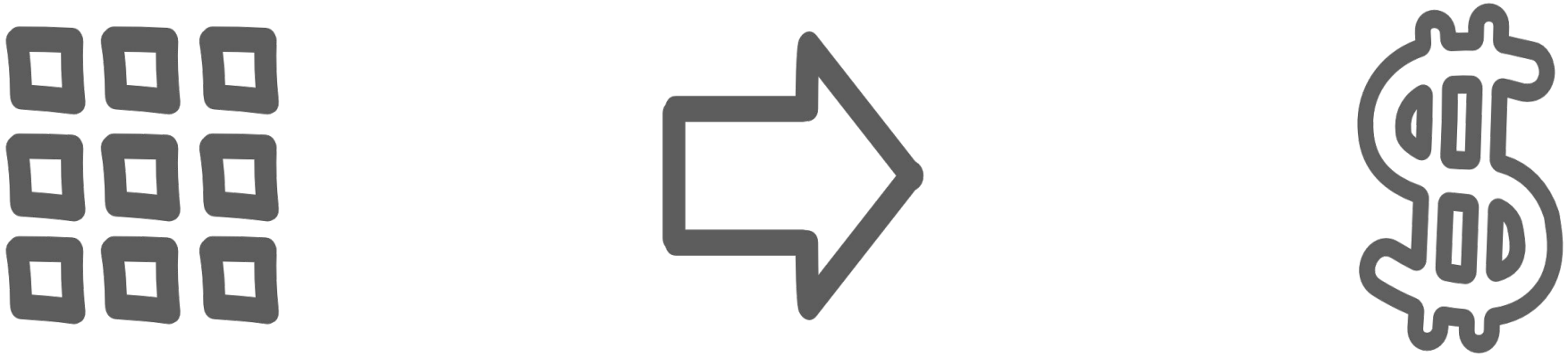
## Framework Features and Ecosystem

Minimize time-to-value by leveraging platform and framework features for common cloud-native requirements such as service discovery, eventing, connectivity, and APIs.

## Operational Efficiency

Footprint and performance are critical decision factors when determining the overall cost of the platform to operate, manage and scale.

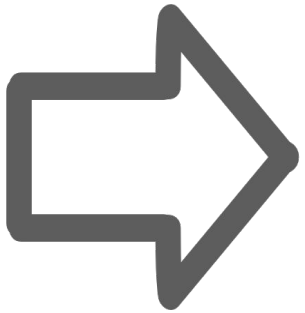
Red Hat

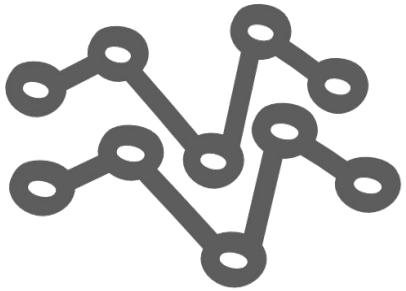


# Supersonic, Subatomic

Improve memory consumption.
Increase deployment density.

Red Hat



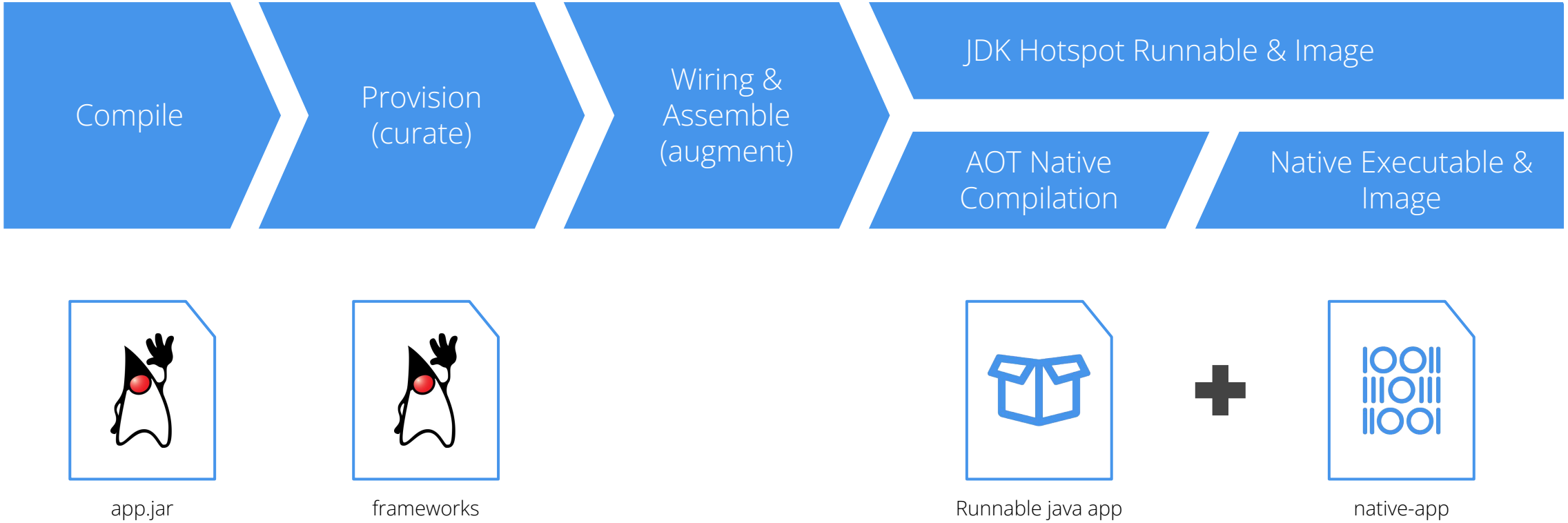


# **Supersonic**, Subatomic

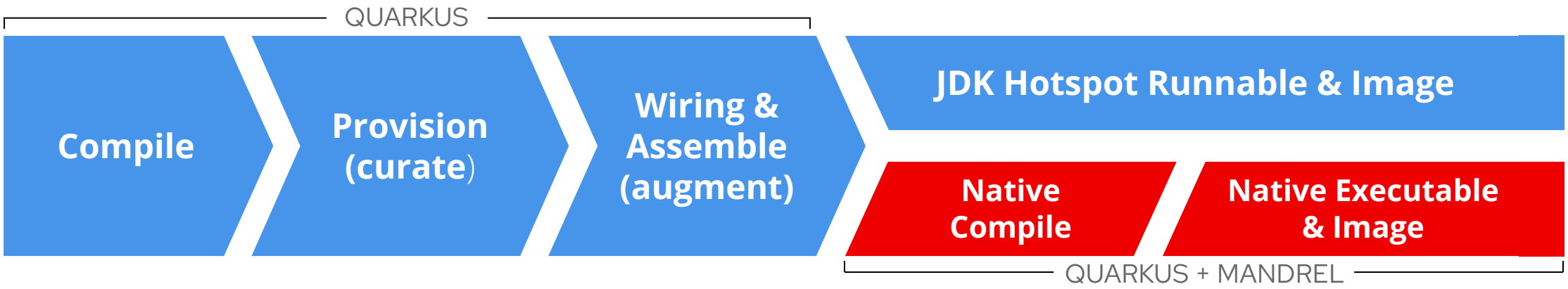
Fast.
Blazing fast to start.
Millisecond fast!

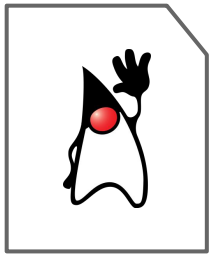
Red Hat

# An ahead-of-time, build-time, runtime

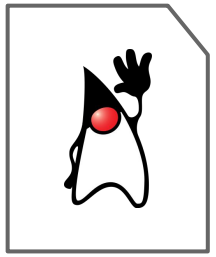


# Quarkus native compilation

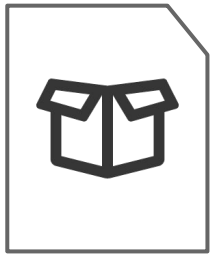


app.jar



Frameworks



Runnable Java app

Runnable Native app

# The Right VM For the Right Deployment

## JIT (OpenJDK HotSpot)

High memory density requirements
High request/s/MB
Fast startup time

Best raw performance (CPU)
Best garbage collectors
Higher heap size usage

Known monitoring tools
Compile Once, Run anywhere
Libraries that only works in standard JDK

## AOT (GraalVM native image)

Highest memory density requirements
Highest request/s/MB
    for low heap size usages
Faster startup time
    10s of ms for Serverless

Red Hat

# Can I Add My Dependencies?

## YES

### Add your own dependency

- Works on the JVM (OpenJDK)
- May work on AOT (GraalVM)

### Write your own extension

- Like add your dependency plus…
- Build time startup and memory improvements
- Better dead code elimination
- Developer Joy

Red Hat

vodafone

"Quarkus seemed to provide the performance boost we needed while at the same time having a good backer (Red Hat) and relying on battle-tested technologies"

**Christos Sotiriou**
DXL technical lead at Vodafone Greece

## Challenge

Running 140 microservices, with heavy spikes in traffic, caused delays and pause while booting new containerized applications leading to waste of marketing efforts.

## Solution

After initial tests indicated that Quarkus would reduce application boot times, reduce CPU and memory usage, and make the entire development process run faster, Vodafone decided to port their most essential libraries and microservices to this new stack.

## Why Quarkus

The main criteria for their selection process to find a replacement for Spring Boot were developer efficiency, lower cloud resource consumption and shorter applications boot-up times. A great impact on cloud resource consumption costs as well as user experience improvement. Their trust of Red Hat combined with its credibility in the software market gave them the assurance that they were making the right choice by selecting Quarkus, whose sponsor is Red Hat.

## Results

- Start-up times have been reduced to almost a quarter without any optimization
- Memory resource consumption was cut in half in JVM mode
- The use of the Quarkus live coding capability (a.k.a. dev mode) resulted in an increase of developer productivity
- Migrating from Spring Boot to Quarkus didn't require a lot of effort for their Spring developers, resulting in a small learning curve
- Far healthier cluster overall, as it is no longer experiencing difficulty in handling the sudden traffic spikes driven by the company's direct marketing campaigns

Source: Vodafone Greece replaces Spring Boot with Quarkus, Nov. 2019.

Red Hat

omnitracs

"The key result from our successful collaboration with Red Hat is how we now approach technology as well as people and processes. We have re-engineered how we develop, test, secure, and deploy software."

Kent Norton
Chief Technology Officer, Omnitracs

## Challenge

Accelerate transformation efforts as the transportation industry is undergoing a digital revolution.

## Solution

Omnitracs partnered with Red Hat to effectively embrace a shift from on-premise development technologies to cloud-native services.

## Why Red Hat

Red Hat was selected based on its reputation for supporting flexible, open technologies in production environments, as well as its knowledge and leadership in enabling cloud-native innovation. The collaboration extended beyond technology development, with Red Hat providing training and strategic guidance on product deployment through Red Hat® Open Innovation Labs.

## Results

- Drove wholesale transformation across the Omnitracs One platform and the entire development process
- Improved reliability, scalability, and overall platform security
- Reduced development time to bring new features to market and more quickly address changing customer needs

## Products and services

Red Hat OpenShift® Container Platform
Red Hat Open Innovation Labs

Source: Red Hat Omnitracs press release, Feb. 2020.

Red Hat

# ALTO Vodafone Case Study (animate me)

- History and background
- Selection criteria
  - Startup time
  - Dev experience
  - Ensure business continuity
- Why Quarkus, alternatives?
  - Java strategy
  - Vert.x, Node.js,
- The journey
  - "When we decided to do the migration, the most important thing was not to break the business continuity"
- Learnings
  - Easy for Spring devs to catch up with Quarkus
    - with Quarkus they saw a *"30 to 40% better developer productivity vis-a-vis Spring Boot, and this is for an ex-Spring Boot developer"*, according to Christos
  - 50-60% memory consumption compared to previous without optimizations
- Current state
  - 80 Quarkus MS in JVM mode
  - 50-60 SB in the maintenance mode

# ALT1 Vodafone Case Study (see speaker notes)

- History and background
- Selection criteria
- Why Quarkus, alternatives?
- The journey
- Learnings
- Current state

# ALT2: Vodafone Case Study 1/2

- History and background
- Selection criteria
  - Startup time
  - Dev experience
  - Ensure business continuity
- Why Quarkus, alternatives?
  - Java strategy
  - Vert.x, Node.js,

# ALT 2 Vodafone Case Study 2/2

- The journey
  - “When we decided to do the migration, the most important thing was not to break the business continuity”
- Learnings
  - Easy for Spring devs to catch up with Quarkus
    - with Quarkus they saw a *“30 to 40% better developer productivity vis-a-vis Spring Boot, and this is for an ex-Spring Boot developer”*, according to Christos
  - 50-60% memory consumption compared to previous without optimizations
- Current state
  - 80 Quarkus MS in JVM mode
  - 50-60 SB in the maintenance mode